ited in the European Search
eport of EP0012 8411.6
Your Ref.: P/2850-43

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 930 747 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
21.07.1999 Bulletin 1999/29

(51) Int. Cl.^{6}: H04L 12/24

(21) Application number: 99250019.9

(22) Date of filing: 20.01.1999

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 20.01.1998 JP 2160198

(71) Applicant: NEC CORPORATION
Tokyo (JP)

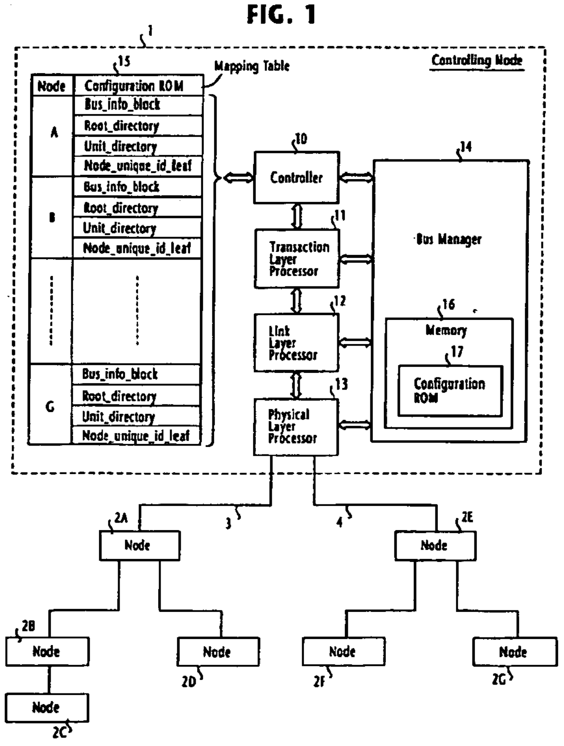
(72) Inventor: Saito, Tomoki
Minato-ku, Tokyo (JP)

(74) Representative:
Patentanwälte Wenzel & Kalkoff
Grubesallee 26
22143 Hamburg (DE)

(54) IEEE 1394 Serial bus system using a mapping table for identifying nodes having required capabilities to establish isochronous connections

(57) In an IEEE-1394 serial bus system, a controlling node reads information from the configuration ROM of each of other nodes attached to the bus and establishes in a mapping table a correspondence between the identifier of each node and the read information. The controlling node identifies particular nodes having required capabilities according to the information stored in the mapping table and requests an isochronous resource manager to obtain information necessary for transmission of isochronous data. The obtained information is then set into plug control registers of the identified nodes to establish a connection between the identified nodes.

FIG. 1



Description

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The present invention relates generally to IEEE 1394 serial bus systems and more specifically to an IEEE 1394 node capable of establishing connections for isochronous transmissions.

Description of the Related Art

[0002] The IEEE-1394 Standard for a High Performance Serial Bus specifies a sequence of different processes to be performed when the bus is initialized. One of such processes is the self identification process in which all nodes attached to the bus are informed of the node identifiers of all other nodes and the maximum speed of each bus segment that is attached between two nodes. However, if it is desired to transfer isochronous data such as video programs between two nodes of particular vendors, details of node information such as node capabilities and their node type are still required in order to identify the nodes supporting isochronous transmission before establishing a connection.

SUMMARY OF THE INVENTION

[0003] It is therefore an object of the present invention to identify nodes having required node capabilities for isochronous transmission and establish a connection between the identified nodes.
[0004] According to the present invention, there is provided an IEEE-1394 serial bus system in which a plurality of nodes are attached to the serial bus, each of the nodes including a configuration ROM and identified by a node identifier. At least one of the nodes comprises a mapping table and control circuitry for reading information from the configuration ROM of each of other nodes and mapping the identifier of each other node to the read information in the mapping table, identifying a node having required node capabilities according to information scored in the mapping table, requesting an isochronous resource manager to obtain information necessary for transmission of isochronous data, and setting the obtained information into the identified node to establish a connection which supports said transmission.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The present invention will be described in further detail with reference to the accompanying drawings, in which:

Fig. 1 shows in block diagram form an IEEE 1394 serial bus tree topology network according to the present invention;
Fig. 2 shows details of a configuration ROM of each node of the tree topology network;
Fig. 3 shows a table mapping process of the present invention at the level of transaction layer;
Figs. 4A to 4D show in flow diagram form the table mapping process of a controlling node;
Figs. 5A to 5D show the formats of packets used in the present invention;
Fig. 6 shows in flow diagram form the operation of the connection establishment process of the controlling node;
Fig. 7 shows information set in plug control registers of audio/visual nodes; and
Fig. 8 shows one sample of connections established among audio/visual nodes of the IEEE 1394 tree topology network.

DETAILED DESCRIPTION

[0006] Referring now to Fig. 1, there is shown an IEEE 1394 serial bus tree-topology network according to the present invention. The network is comprised of a plurality of IEEE 1394 devices, or nodes. One of the nodes is a controlling node 1 and the other nodes are controlled nodes, or target nodes 2A to 2G attached to the controlling node by local buses 3 and 4 in a tree-like topology. Nodes 1 and 2 may be audio/visual devices or personal computers, or computer peripheral devices, all of which are designed to the IEEE Standard for a High Performance Serial Bus (or IEEE Std 1394-1995) which supports both asynchronous and isochronous transmissions, simultaneously.
[0007] Controlling node 1 includes a controller 10, a transaction layer processor 11, a link layer processor 12 and a physical layer processor 13, all of which are connected to a bus manager 14. By way of the processors 11 to 13, the controller 10 transmits and receives signals to and from the local buses 3 and 4. Further associated with the controller 10 is a mapping table 15 in which the controller 10 maps node identifiers to the corresponding device information items obtained from the configuration ROM of target nodes in a manner as will be described in detail later.
[0008] A memory 16 is provided within the bus manager 14 and a configuration ROM 17 is defined within the memory 16. Each of target nodes 2A to 2G has its own configuration ROM.
[0009] As shown in detail in Fig. 2, the memory 16 of the controlling node as well as target nodes 2A to 2G is partitioned into a plurality of register spaces including the space for the configuration ROM 17, a space for CSR (control and status register) 18, and initial units space 19. Bus manager 14 controls the layers 11, 12 and 13 according to the contents of the CSR memory space 18. After the mapping table 15 is completed, the controller 10 exchanges command messages with other

nodes using addresses specified in the CSR memory space 18.

[0010] Configuration ROM 17 is divided into a number of storage locations each with a 32-bit (quadlet) wide memory space. These quadlet storage locations are grouped into sections such as bus_info_block, root_directory, unit_directories and a node_unique_id_leaves, and each section begins with a memory space in which the data length of the section and error check data are indicated.

[0011] Audio/visual nodes are provided with master plug registers and plug control registers designed to the IEC standard 61883 for digital interface for consumer electronic audio/video equipment (reference number 100C/46 ~ 50/CVD, project number 100C/61883-1~5/Ed.1). A maximum of thirty-two plug control registers are provided in each of the target nodes 2. In each of these plug control registers are stored information relating to the presence/absence of broadcast connection, the number of point-to-point connections, the identification of an isochronous channel and the transfer rate and bandwidth of the isochronous channel. Management of the plug control registers is provided by the master plug registers. Plug control registers and master plug registers is defined within the initial units space 19 as illustrated in Fig. 2.

[0012] Controlling node 1 begins its operation with a bus initialization process by asserting a bus reset on local buses 3 and 4. Following the bus initialization, a tree identification process begins to identify the root node and the isochronous resource manager and the topology of all attached nodes, resulting in all ports of the nodes being designated as either child or parent ports. A child port connects to a node further away from the root, while a parent port connects to a node closer to the root. The tree identification process is then followed by a self identification process during which a physical identification is assigned to each node, neighboring nodes exchange their speed capabilities and the topology defined during tree identification.

[0013] Following the self identification process, the controller 10 performs a mapping process on the table 15. The cable mapping process of controller 10 begins at the transaction layer with a target node, 2A, for example, to establish correspondences in the mapping table 15 between the identifier of the node 2A and a number of its unique device information items. As illustrated, the controller 10 at node 1 sends a transaction request to its transaction layer 11 to forward a read request packet to the transaction layer 11A of node 2A, which replies with an acknowledgment packet and sends a transaction indication to its configuration ROM 17A. This results in a desired data item being read out of the configuration ROM 17A and transmitted as a transaction response to the transaction layer 11A. The configuration ROM data is placed in the data field of a read response packet and sent from the transaction layer 11A to the transaction layer 11, which returns an acknowledgment packet to node 2A and sends a transaction indication to the controller 10. The configuration ROM data of node 2A contained in the transaction indication is mapped in the mapping table 15 to the identifier of node 2A.

[0014] The following is a detailed description of the table mapping operation of controller 10 with the aide of the flowcharts of Figs. 4A to 4D by using the standardized packets as shown in Figs. 5A to 5D.

[0015] The table mapping process begins with the reading of bus_into_block section data from all target nodes. At step 101, the controller 10 formulates a Read Data Quadlet (RDQ) unicast request packet (Fig. 5A) by setting $3FF_{16}$ (=1023) in the ten most significant bits of the destination ID field to specify all local buses and setting one of 0_{16} to $3E_{16}$ in the six least significant bits and an address ($FFFFF0000400_{16}$) in the destination_offset field indicating the first quadlet data of bus_info_block section. In this way, RDQ unicast packets can be individually addressed to a maximum of sixty-three target nodes.

[0016] All nodes recognizing a receipt of the RDQ request packet know that they are being targeted and respond with a RDQ response packet which is shown in Fig. 5B. More specifically, each target node reads the first quadlet (four bytes) data of bus_info_block section of its configuration ROM as specified by the destination_offset field of the received packet and inserts this first quadlet data into the quadlet_data field of the RDQ response packet as well as the node ID of the target node into its source_ID field. This first quadlet data includes the data length of the bus_info_block section and CRC information (see Fig. 2).

[0017] In response to receipt of the RDQ response packet from each target node (step 103), the controller 10 proceeds to step 104 to save the source address contained in its source_ID field and reads the length data of bus_info_block section from the quadlet data field.

[0018] At step 105, the controller 10 checks to see if RDQ response packets are received from all target nodes. If not, flow returns to step 101 to repeat the process until RDQ response packets are received from all target nodes.

[0019] When the decision at step 105 becomes affirmative, flow proceeds to step 106 to add one quadlet to the initial offset data to produce a summed address value $FFFFF0000404_{16}$ which indicates the second address location of the bus_info_block section and subtract one quadlet from the length data saved at step 104.

[0020] At step 107, the controller 10 formulates a unicast Read Data Block (RDB) request packet (see Fig. 5C) for a target node by setting the saved identifier of the node in the destination ID field of the packet, the summed address value in the destination_offset field and the subtracted length data in the data_length field, the packet being forwarded onto all local buses (step 108) so that only one node specified by the destination

ID field knows that it is being targeted.

[0021] In response to the RDB request packet, the target node replies with a RDB response packet (Fig. 5D) by setting its data field with bus_info_block section data that begins with the second address location specified by the request packet.

[0022] Upon receipt of the RDB response packet from a target node (step 109), the controller 10 proceeds to step 110 to establish a correspondence in the mapping table 15 between the target node identifier and the bus_info_block section data contained in the response packet.

[0023] At step 111, the controller 10 checks to see if RDB response packets are received from all target nodes. If not, flow returns to step 106 to repeat the process until RDB response packets are received from all target nodes.

[0024] Following the affirmative decision at step 111, flow proceeds to step 201 (Fig. 4B) to start reading root_directory section data from the target nodes in a manner similar to the routine of Fig. 4A by formulating a unicast RDQ request packet. In this unicast request packet all-bus address $3FF_{16}$ is set in the ten most significant bits of the destination ID field and an individual address is set in the six least significant bits similar to step 101 and the first address ($FFFFF0000414_{16}$) of the root_directory section is set in the destination_offset field.

[0025] The RDQ request packet is forwarded to all local buses (step 202) so that all nodes know that they are being targeted and reply with RDQ response packets each containing the data length of the root_directory section in its quadlet data field.

[0026] Upon receipt of the RDQ response packet from each target node (step 203), the controller 10 proceeds to step 204 to save the source identifier and the length data of the root_directory section.

[0027] At step 205, the controller 10 checks to see if RDQ response packets are received from all target nodes. If not, the flow returns to step 201 to repeat the process until RDQ response packets are received from all target nodes.

[0028] When the decision at step 205 becomes affirmative, flow proceeds to step 206 to add one quadlet to the initial offset data to produce a summed address indicating the second address location of the root_directory section and subtract one quadlet from the length data saved at step 204.

[0029] At step 207, the controller 10 formulates a unicast RDB request packet for a target node by setting the saved identifier of the node in the destination ID field of the packet, the summed address value in the destination_ offset field and the subtracted length data in the data_length field. The packet is forwarded onto all local buses (step 208) so that only one node specified by the destination ID field knows that it is being targeted.

[0030] In response to the RDB request packet, the target node replies with a RDB response packet containing in its data field root_directory section data that begins with the second address location specified by the request packet.

[0031] Upon receipt of the RDB response packet from a target node (step 209), the controller 10 proceeds to step 210 to establish a correspondence in the mapping table 15 between the identifier of the target node and the root_directory section data contained in the response packet.

[0032] At step 211, the controller saves the unit_directory offset data and node_unique_id_leaf offset data contained in the data field of the response packet, and proceeds to step 212 to check to see if RDB response packets are received from all target nodes. If not, flow returns to step 206 to repeat the process until RDB response packets are received from all target nodes.

[0033] Following the affirmative decision at step 212, flow proceeds to step 301 (Fig. 4C) to start reading the unit_directory section data from each target node by formulating a unicast RDQ request packet. In this unicast request packet the identifier of a node saved at step 211 is set in the destination ID field and the unit_directory offset data saved at step 211 is set in the destination_offset field. The unicast packet is forwarded onto the local buses at step 302, so that only one node knows that it is being targeted. The target node replies with a RDQ response packet containing the length data of the unit_directory section.

[0034] When the controlling node 1 receives the RDQ response packet (step 303), flow proceeds to step 304 to save the node identifier and length data and proceeds to step 305 to check to see if RDQ response packets are received from all target nodes. If not, flow returns to step 301 to repeat the process until RDQ response packets are received from all target nodes.

[0035] When the decision at step 305 becomes affirmative, flow proceeds to step 306 to add one quadlet to the unit_directory offset data of a given node saved at step 304 to specify the second address location of unit_directory section and subtract one quadlet from the length data saved at step 211.

[0036] At step 307, the controller 10 formulates a RDB request packet for the given node by setting its saved identifier in the destination ID field, summed address in the destination_offset field and the subtracted length data in the data_length field. At step 308, the RDB request packet is forwarded onto the local buses.

[0037] In response to the RDB request packet, the given target node replies with a RDB response packet containing in its data field unit_directory section data.

[0038] When the controller 10 receives the RDB response packet at step 309, it proceeds to step 310 to establish a correspondence in the mapping table 15 between the identifier of the given node and the unit_directory section data contained in the packet. At step 311, the controller 10 checks to see if RDB response packets are received from all target nodes. If not, steps 306 to 310 are repeated until RDB response packets are received from all nodes.

[0039] Following the affirmative decision at step 311, flow proceeds to step 401 (Fig. 4D) to start reading node_unique_ id_leaf section data from each target node by formulating a unicast RDQ request packet. In this unicast request packet the identifier of a node saved at step 211 is set in the destination ID field and the node_unique_id_leaf offset data saved at step 211 is set in the destination_offset field. The unicast packet is forwarded onto the local buses at step 402, so that only one node knows that it is being targeted. The target node replies with a RDQ response packet containing the length data of the node_unique_id_leaf section.

[0040] In response to receipt of the RDQ response packet (step 403), flow proceeds to step 404 to save the node identifier and length data and proceeds to step 405 to check to see if RDQ response packets are received from all target nodes. If not, flow returns to step 401 to repeat the process until RDQ response packets are received from all target nodes,

[0041] When the decision at step 405 becomes affirmative, flow proceeds to step 406 to add one quadlet to the node_unique_id_leaf offset data of a given node saved at step 404 to specify the second address location of node_unique_id_leaf section and subtract one quadlet from the length data saved at step 211.

[0042] At step 407, the controller 10 formulates a RDB request packet for the given node by setting its saved identifier in the destination ID field, summed address in the destination_offset field and the subtracted length data in the data_length field. At step 408, the RDB request packet is forwarded onto the local buses.

[0043] In response to the RDB request packet, the given target node replies with a RDB response packet containing in its data field node_unique_id_leaf section data.

[0044] When the controller 10 receives the RDB response packet at step 409, it proceeds to step 410 to establish a correspondence in the mapping table 15 between the identifier of the given node and the unit_directory section data contained in the packet. At step 411, the controller 10 checks to see if RDB response packets are received from all target nodes. If not, steps 406 to 410 are repeated until RDB response packets are received from all nodes, whereupon all routines of the table mapping process is terminated.

[0045] With the mapping table 15 being completed as shown in Fig. 1, the controlling node 1 starts a process for establishing a connection between two audio/visual nodes.

[0046] As shown in Fig. 6, the connection establishment process begins with step 61 in which the controller 10 reads information from all entries of the mapping table 15 and identifies nodes having isochronous transmission capabilities. At step 62, the controller 10 sends an asynchronous lock packet to the isochronous resource manager to request the use of a bandwidth. If the request is granted, the isochronous resource manager returns a response packet indicating a channel number and a bandwidth, The controller 10 sets the requested information and other necessary information into the input and output plug control registers of the identified nodes as shown in Fig. 7 according to the IEC-61883 standard. At step 63, the controller 10 commands the output plug control register to establish a connection to the input plug control register according to the set information. In this way, an isochronous channel is established between audio/visual nodes for transmission of isochronous data.

[0047] Fig. 8 shows one example of connections established among audio/visual nodes of the present invention. It is seen that a plurality of connections are established on a single isochronous channel. One point-to-point connection 81 is established between the output plug control register 91 of node 71 and the input plug control register 95 of node 75, two point-to-point connections 82 are established between the output PCR 91 and the input PCR 93 of node 73, and three point-to-point connections 83 are established between the output PCR 91 and the input PCR 92 of node 72. Additionally, a broadcast-out connection 84 is established between the output PCR 91 and the broadcast channel number of the isochronous channel and a broadcast-in connection 85 is established between the input PCR 95 and the broadcast channel number, These broadcast connections are established independently of the operating states of the transmit and receive nodes. The plug control registers of the broadcast connections can be rewritten from any other nodes of the network, so that an established broadcast connection can be cleared and the ownership of the connection be transferred to another node.

[0048] With a connection being established between two audio/visual nodes, start/stop, pause and slow-motion commands (AV/C commands) are used to control video program transmissions according to the function control protocol (FCP) of the IEC-61883 standard (step 64). At the end of the isochronous transmission (step 65), the input and output plug control registers of the audio/visual nodes are reset to release the connection (step 66).

Claims

1. A node attached to an IEEE-1394 serial bus for controlling devices attached to said bus, each of said devices including a configuration ROM and identified by an identifier, said node comprising:

    a mapping cable; and
    control circuitry for reading information from the configuration ROM of each of said devices and mapping the identifier of the device to the read information in said mapping table, identifying a

device having required capabilities according to information stored in said mapping table, requesting an isochronous resource manager to obtain information necessary for transmission of isochronous data, and setting the obtained information into said identified device to establish a connection which supports said transmission.

2. An IEEE-1394 serial bus system in which a plurality of nodes are attached to the serial bus, each of said nodes including a configuration ROM and identified by a node identifier, at least one of said nodes comprising:

   a mapping table; and
   control circuitry for reading information from the configuration ROM of each of other nodes and mapping the identifier of each other node to the read information in said mapping table, identifying a node having required node capabilities according to information stored in said mapping table, requesting an isochronous resource manager to obtain information necessary for transmission of isochronous data, and setting the obtained information into said identified node to establish a connection which supports said transmission.

3. A method for establishing a connection between ones of a plurality nodes attached to an IEEE-1394 serial bus, each of said nodes including a configuration ROM and identified by a node identifier, comprising the steps of:

   a) reading information from the configuration ROM of each of said nodes and mapping the identifier of the node to the read information in a mapping table;
   b) identifying a node having required node capabilities according to information stored in said mapping table;
   c) requesting an isochronous resource manager to obtain information necessary for transmission of isochronous data; and
   d) establishing a connection to said identified node according to the obtained information.

4. The method of claim 3, wherein the step (a) comprises the steps of:

   reading a data quadlet from the configuration ROM of each node; and
   reading a data block from a location of the configuration ROM of the node which is specified by said data quadlet and mapping the data block to the node identifier of the node in said mapping table.

5. The method of claim 3, wherein the step (a) comprises the steps of:

   transmitting a read data quadlet (RDQ) unicast request packet to the serial bus indicating a storage location of a data quadlet and receiving a (RDQ) response packet from each of said nodes indicating the data length of a data block following said data quadlet; and
   transmitting a read data block (RDB) unicast request packet to the serial bus indicating said data length and a storage location of said data block and receiving a RDB response packet from each node so that said data block is obtained from the configuration ROM of the node and mapping the obtained data block to the node identifier of the node in said mapping table.

6. The method of claim 3, wherein said identified node includes a plug control register, wherein step (b) comprises:

   reading data from the mapping cable to identify ones of said nodes having required node capabilities;
   requesting an isochronous resource manager to obtain information necessary for transmission of isochronous data;
   setting the obtained information into the plug control register of said identified node; and
   establishing a connection according to the information set in said plug control register.

FIG. 1

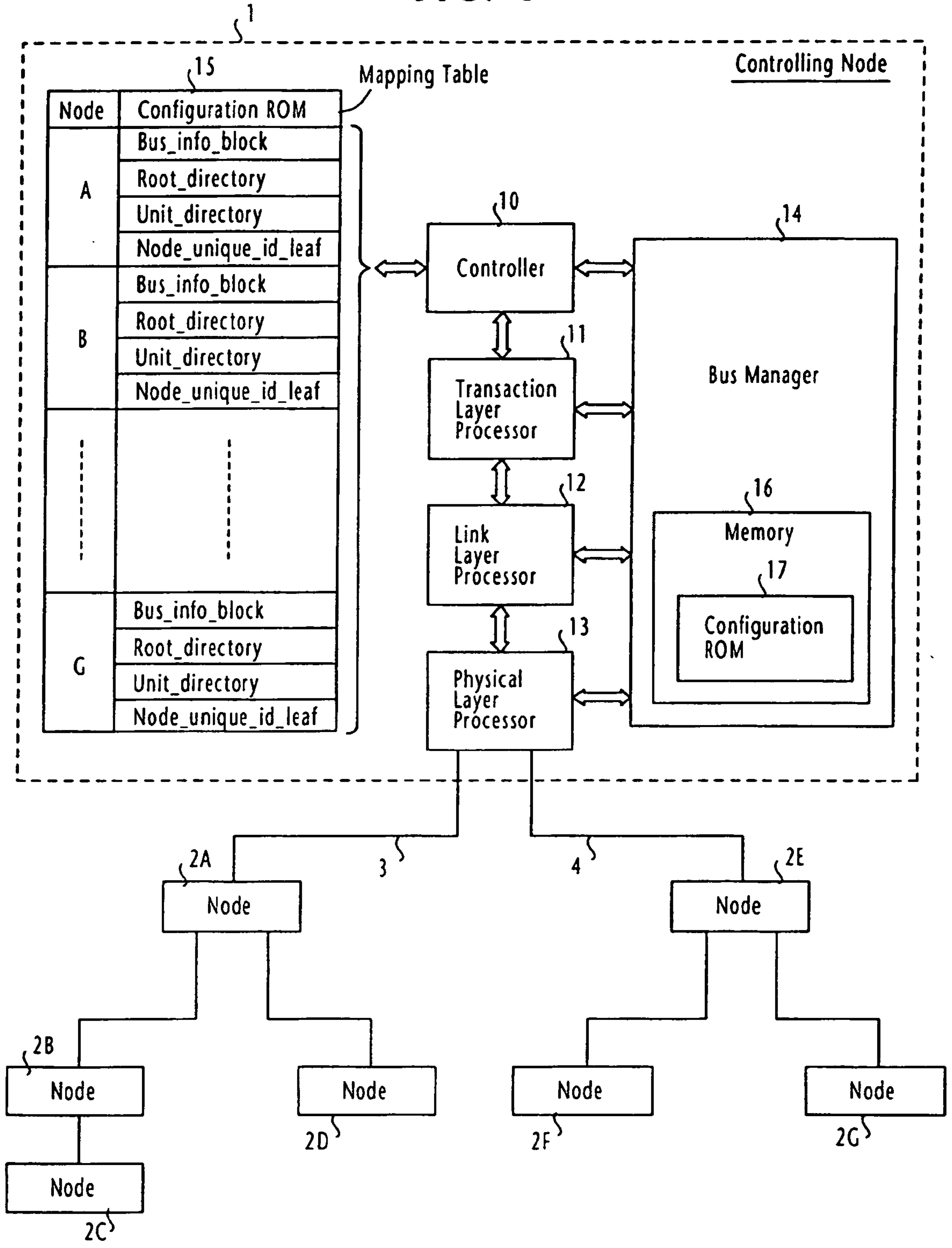

1
Controlling Node
15
Mapping Table
Node
Configuration ROM
A
Bus_info_block
Root_directory
Unit_directory
Node_unique_id_leaf
B
Bus_info_block
Root_directory
Unit_directory
Node_unique_id_leaf
G
Bus_info_block
Root_directory
Unit_directory
Node_unique_id_leaf
10
Controller
11
Transaction Layer Processor
12
Link Layer Processor
13
Physical Layer Processor
14
Bus Manager
16
Memory
17
Configuration ROM
3
4
2A
Node
2E
Node
2B
Node
Node
2D
Node
2F
Node
2G
Node
2C

FIG. 2

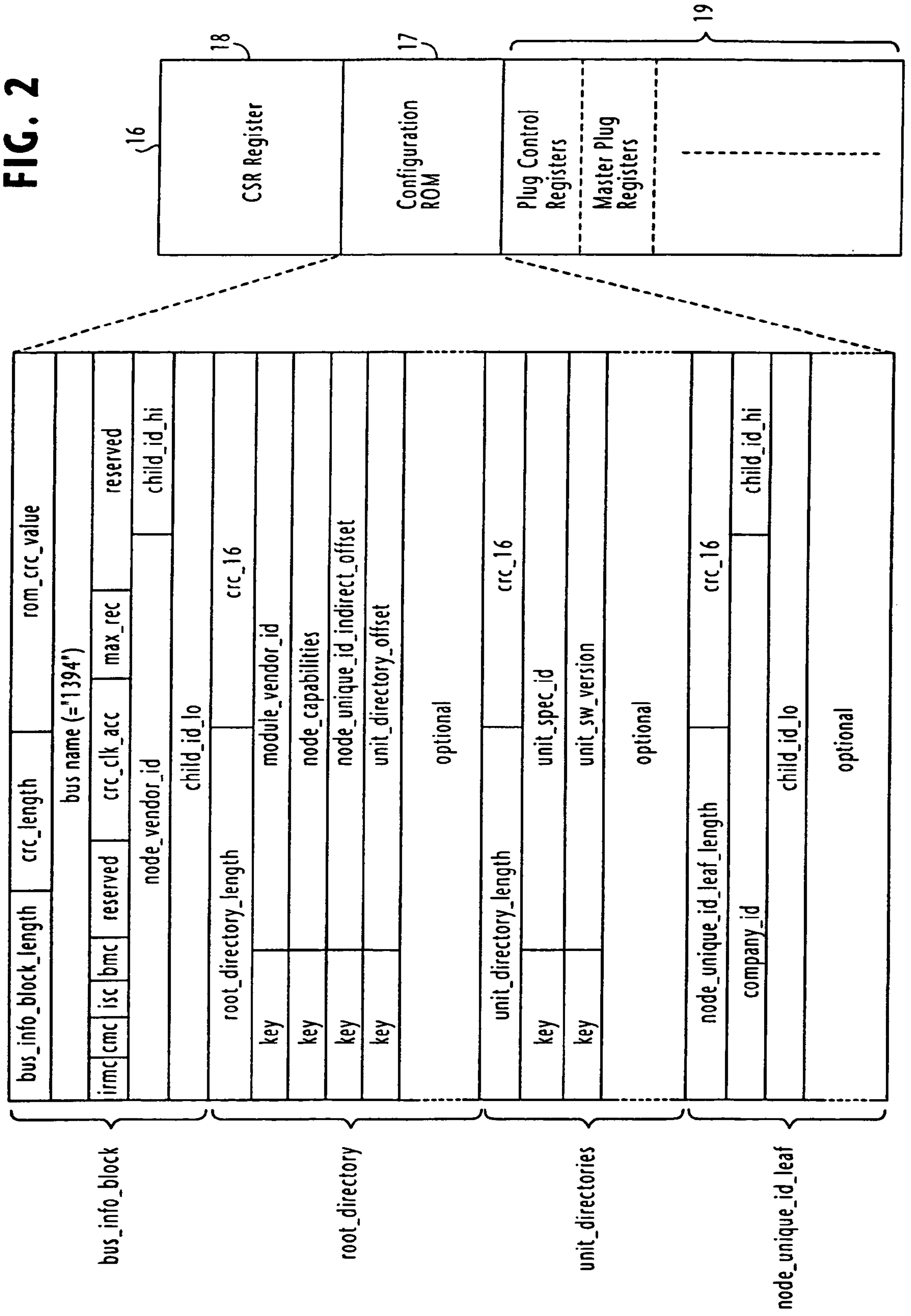

16
18
17
19
CSR Register
Configuration ROM
Plug Control Registers
Master Plug Registers
bus_info_block_length
crc_length
rom_crc_value
bus name (="1394")
irmc
cmc
isc
bmc
reserved
crc_clk_acc
max_rec
reserved
node_vendor_id
child_id_hi
child_id_lo
root_directory_length
crc_16
key
module_vendor_id
key
node_capabilities
key
node_unique_id_indirect_offset
key
unit_directory_offset
optional
unit_directory_length
crc_16
key
unit_spec_id
key
unit_sw_version
optional
node_unique_id_leaf_length
crc_16
company_id
child_id_hi
child_id_lo
optional
bus_info_block
root_directory
unit_directories
node_unique_id_leaf

FIG. 3

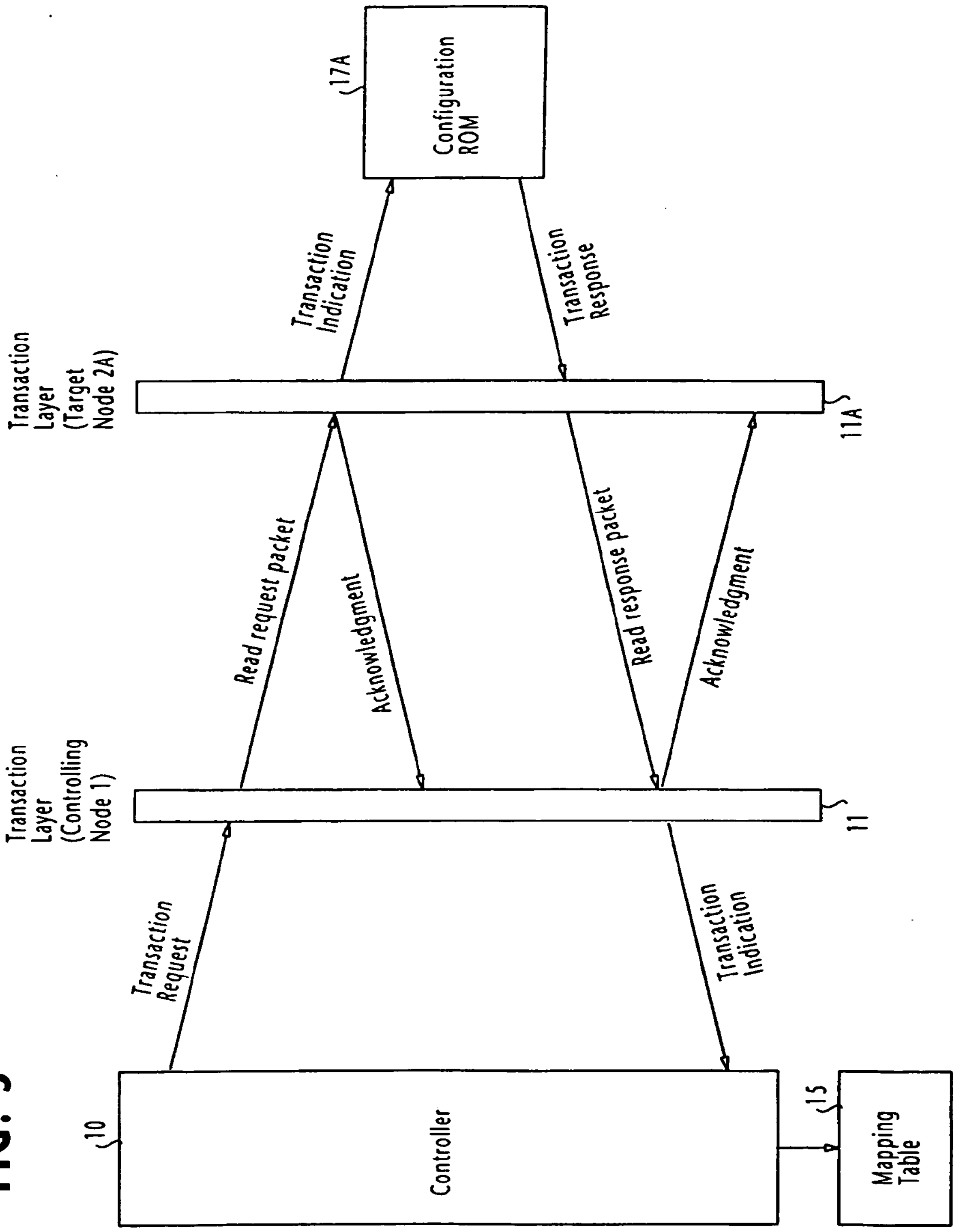

Configuration ROM
17A
Transaction Indication
Transaction Response
Transaction Layer (Target Node 2A)
11A
Read request packet
Acknowledgment
Read response packet
Acknowledgment
Transaction Layer (Controlling Node 1)
11
Transaction Request
Transaction Indication
Controller
10
Mapping Table
15

FIG. 4A

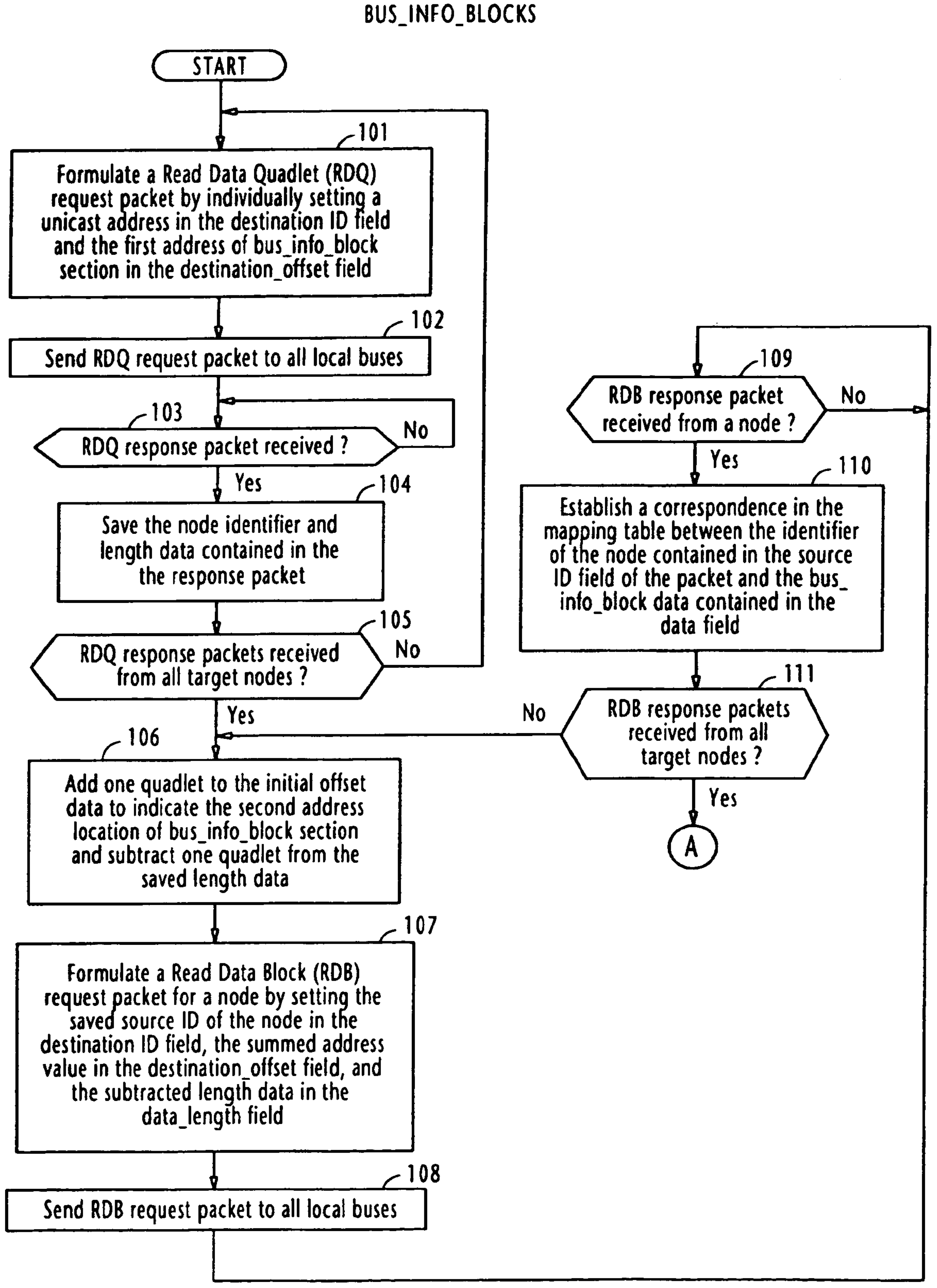

BUS_INFO_BLOCKS
START
101
Formulate a Read Data Quadlet (RDQ) request packet by individually setting a unicast address in the destination ID field and the first address of bus_info_block section in the destination_offset field
102
Send RDQ request packet to all local buses
103
RDQ response packet received ?
No
Yes
104
Save the node identifier and length data contained in the the response packet
105
RDQ response packets received from all target nodes ?
No
Yes
106
Add one quadlet to the initial offset data to indicate the second address location of bus_info_block section and subtract one quadlet from the saved length data
107
Formulate a Read Data Block (RDB) request packet for a node by setting the saved source ID of the node in the destination ID field, the summed address value in the destination_offset field, and the subtracted length data in the data_length field
108
Send RDB request packet to all local buses
109
RDB response packet received from a node ?
No
Yes
110
Establish a correspondence in the mapping table between the identifier of the node contained in the source ID field of the packet and the bus_info_block data contained in the data field
111
RDB response packets received from all target nodes ?
No
Yes
A

FIG. 4B

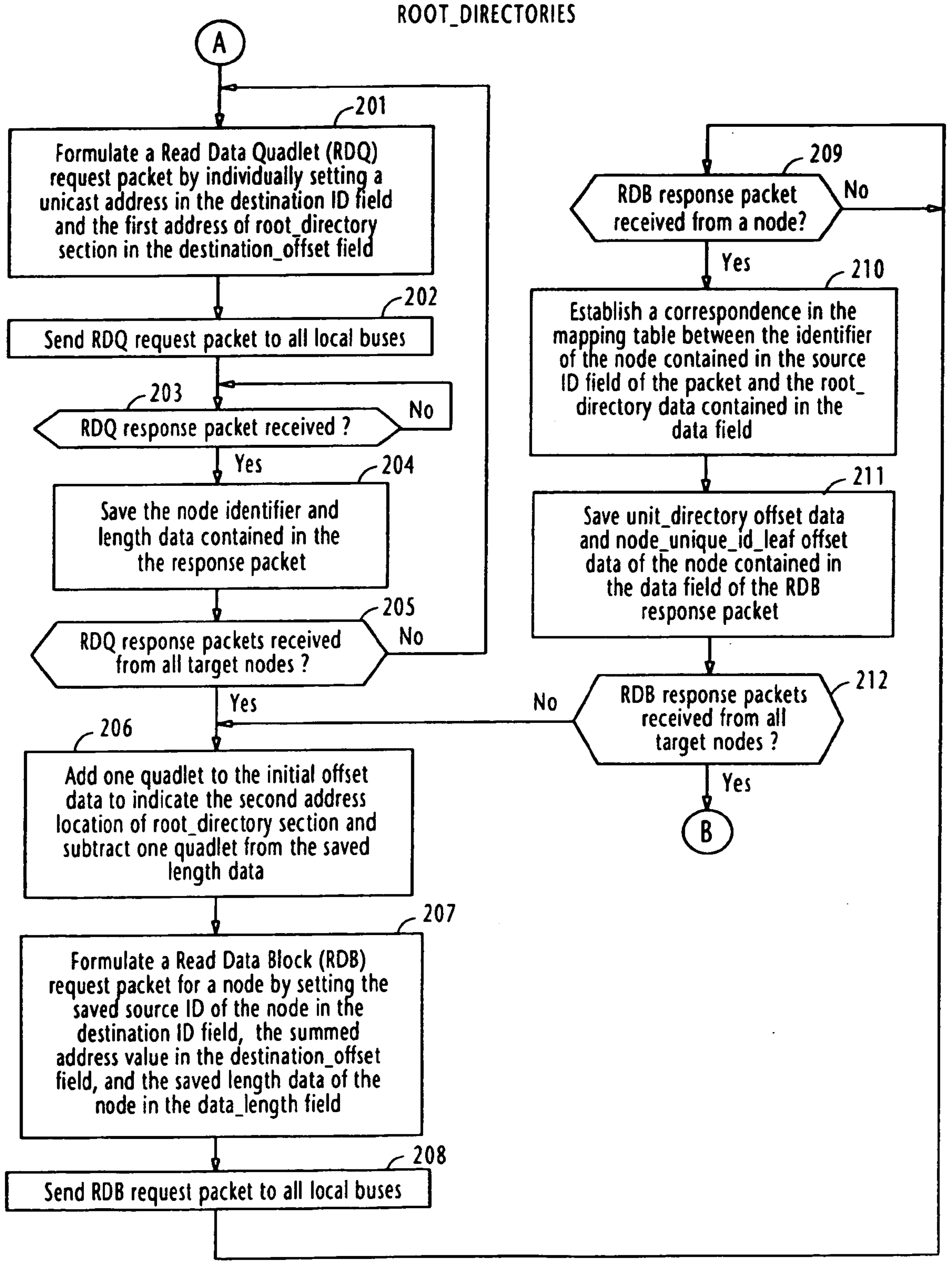

ROOT_DIRECTORIES
A
201
Formulate a Read Data Quadlet (RDQ) request packet by individually setting a unicast address in the destination ID field and the first address of root_directory section in the destination_offset field
202
Send RDQ request packet to all local buses
203
RDQ response packet received ?
No
Yes
204
Save the node identifier and length data contained in the the response packet
205
RDQ response packets received from all target nodes ?
No
Yes
206
Add one quadlet to the initial offset data to indicate the second address location of root_directory section and subtract one quadlet from the saved length data
207
Formulate a Read Data Block (RDB) request packet for a node by setting the saved source ID of the node in the destination ID field, the summed address value in the destination_offset field, and the saved length data of the node in the data_length field
208
Send RDB request packet to all local buses
209
RDB response packet received from a node?
No
Yes
210
Establish a correspondence in the mapping table between the identifier of the node contained in the source ID field of the packet and the root_ directory data contained in the data field
211
Save unit_directory offset data and node_unique_id_leaf offset data of the node contained in the data field of the RDB response packet
212
RDB response packets received from all target nodes ?
No
Yes
B

FIG. 4C

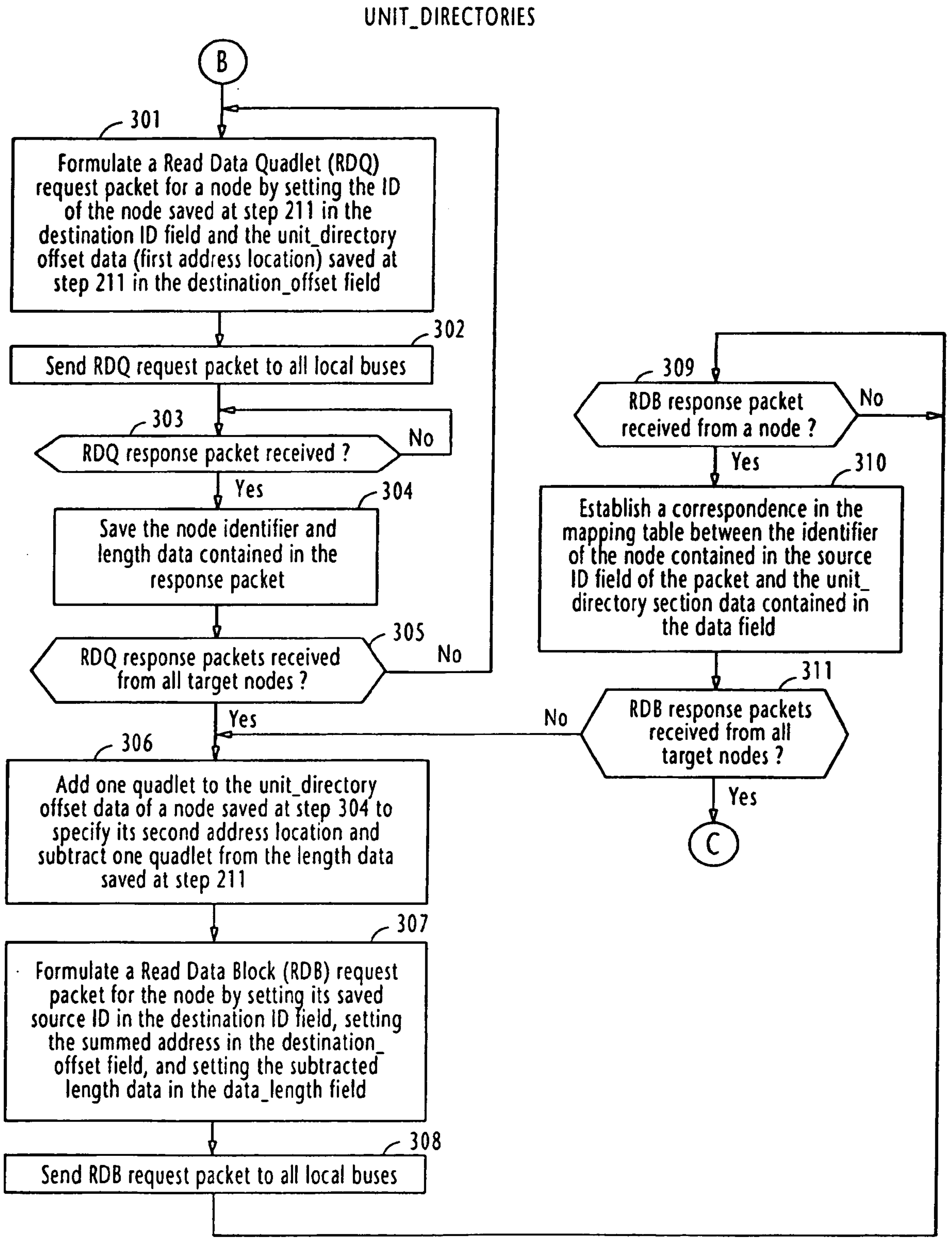

UNIT_DIRECTORIES
B
301
Formulate a Read Data Quadlet (RDQ) request packet for a node by setting the ID of the node saved at step 211 in the destination ID field and the unit_directory offset data (first address location) saved at step 211 in the destination_offset field
302
Send RDQ request packet to all local buses
303
RDQ response packet received ?
No
Yes
304
Save the node identifier and length data contained in the response packet
305
RDQ response packets received from all target nodes ?
No
Yes
306
Add one quadlet to the unit_directory offset data of a node saved at step 304 to specify its second address location and subtract one quadlet from the length data saved at step 211
307
Formulate a Read Data Block (RDB) request packet for the node by setting its saved source ID in the destination ID field, setting the summed address in the destination_offset field, and setting the subtracted length data in the data_length field
308
Send RDB request packet to all local buses
309
RDB response packet received from a node ?
No
Yes
310
Establish a correspondence in the mapping table between the identifier of the node contained in the source ID field of the packet and the unit_directory section data contained in the data field
311
RDB response packets received from all target nodes ?
No
Yes
C

FIG. 4D

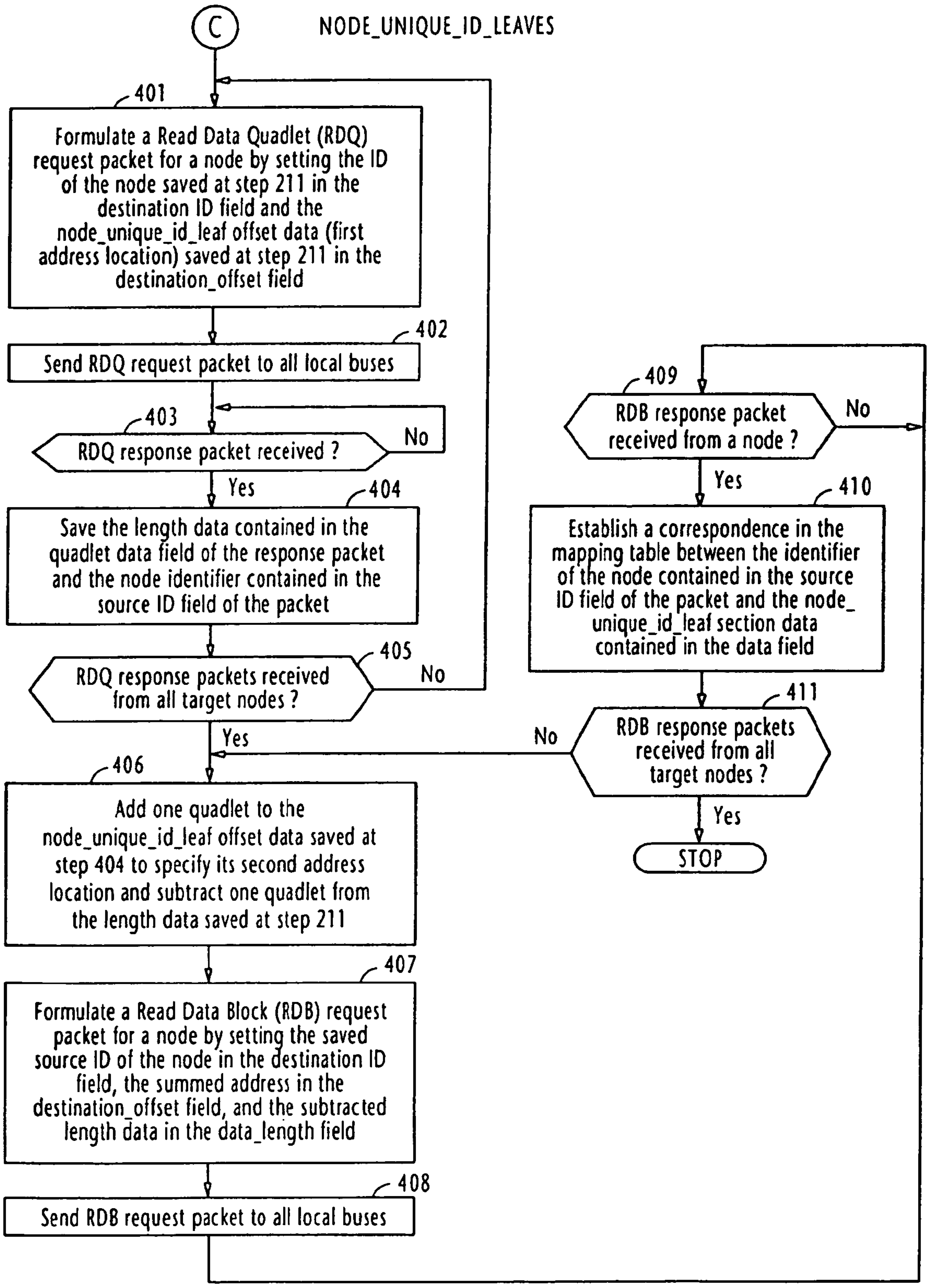

C
NODE_UNIQUE_ID_LEAVES
401
Formulate a Read Data Quadlet (RDQ) request packet for a node by setting the ID of the node saved at step 211 in the destination ID field and the node_unique_id_leaf offset data (first address location) saved at step 211 in the destination_offset field
402
Send RDQ request packet to all local buses
403
RDQ response packet received ?
No
Yes
404
Save the length data contained in the quadlet data field of the response packet and the node identifier contained in the source ID field of the packet
405
RDQ response packets received from all target nodes ?
No
Yes
406
Add one quadlet to the node_unique_id_leaf offset data saved at step 404 to specify its second address location and subtract one quadlet from the length data saved at step 211
407
Formulate a Read Data Block (RDB) request packet for a node by setting the saved source ID of the node in the destination ID field, the summed address in the destination_offset field, and the subtracted length data in the data_length field
408
Send RDB request packet to all local buses
409
RDB response packet received from a node ?
No
Yes
410
Establish a correspondence in the mapping table between the identifier of the node contained in the source ID field of the packet and the node_ unique_id_leaf section data contained in the data field
411
RDB response packets received from all target nodes ?
No
Yes
STOP

FIG. 5A

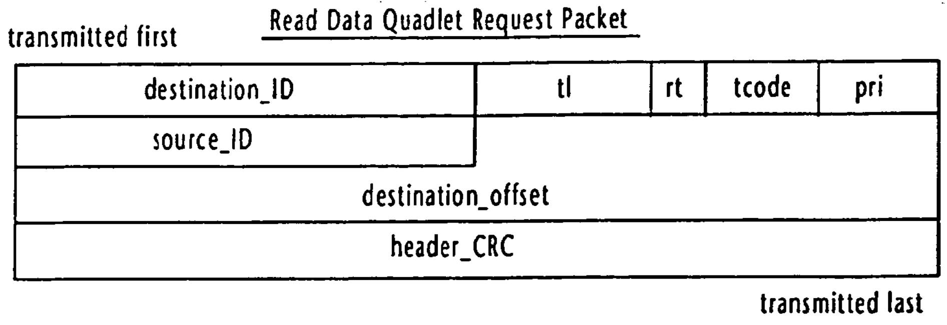


FIG. 5B

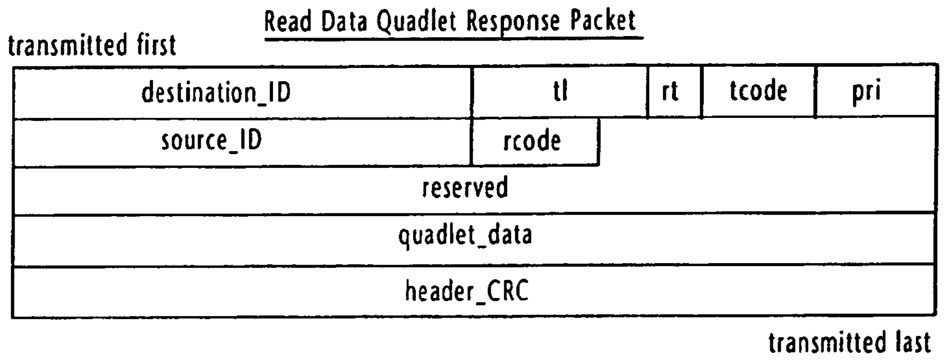


FIG. 5C

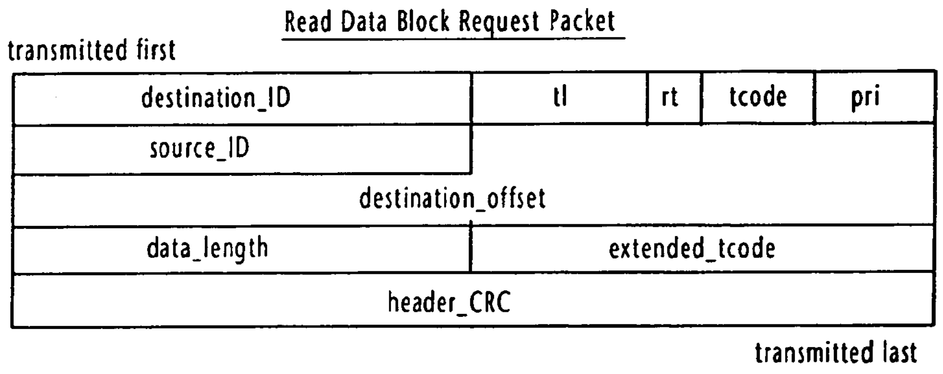


FIG. 5D

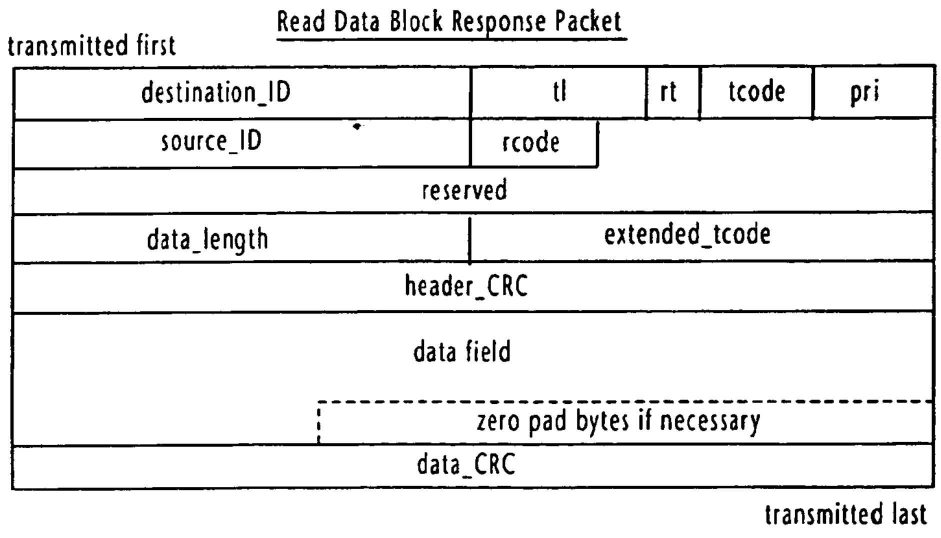


FIG. 6

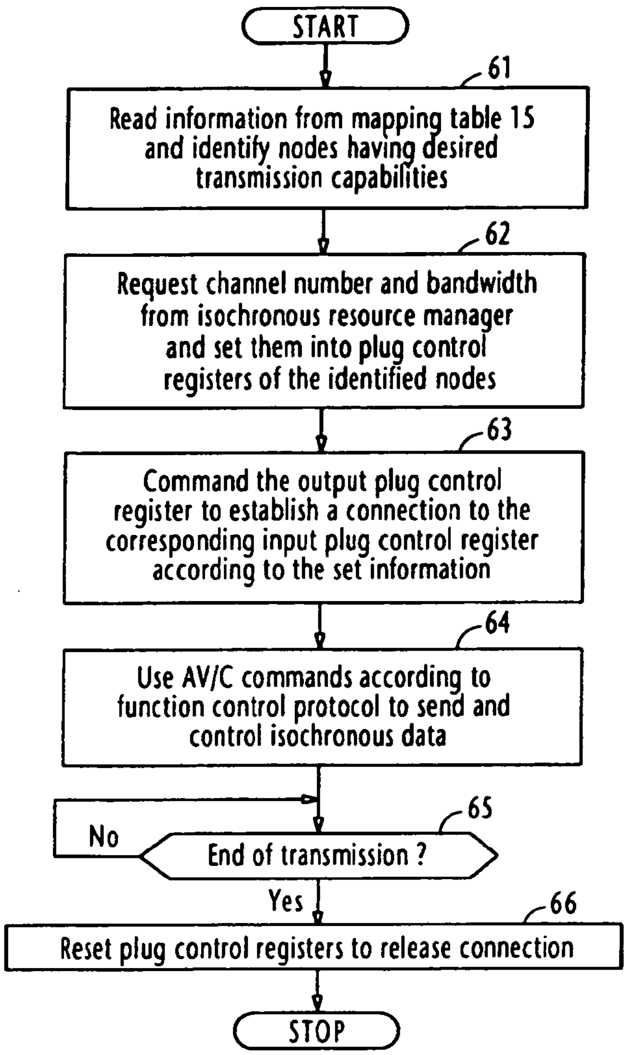

START
61
Read information from mapping table 15 and identify nodes having desired transmission capabilities
62
Request channel number and bandwidth from isochronous resource manager and set them into plug control registers of the identified nodes
63
Command the output plug control register to establish a connection to the corresponding input plug control register according to the set information
64
Use AV/C commands according to function control protocol to send and control isochronous data
65
No
End of transmission ?
Yes
66
Reset plug control registers to release connection
STOP

FIG. 7

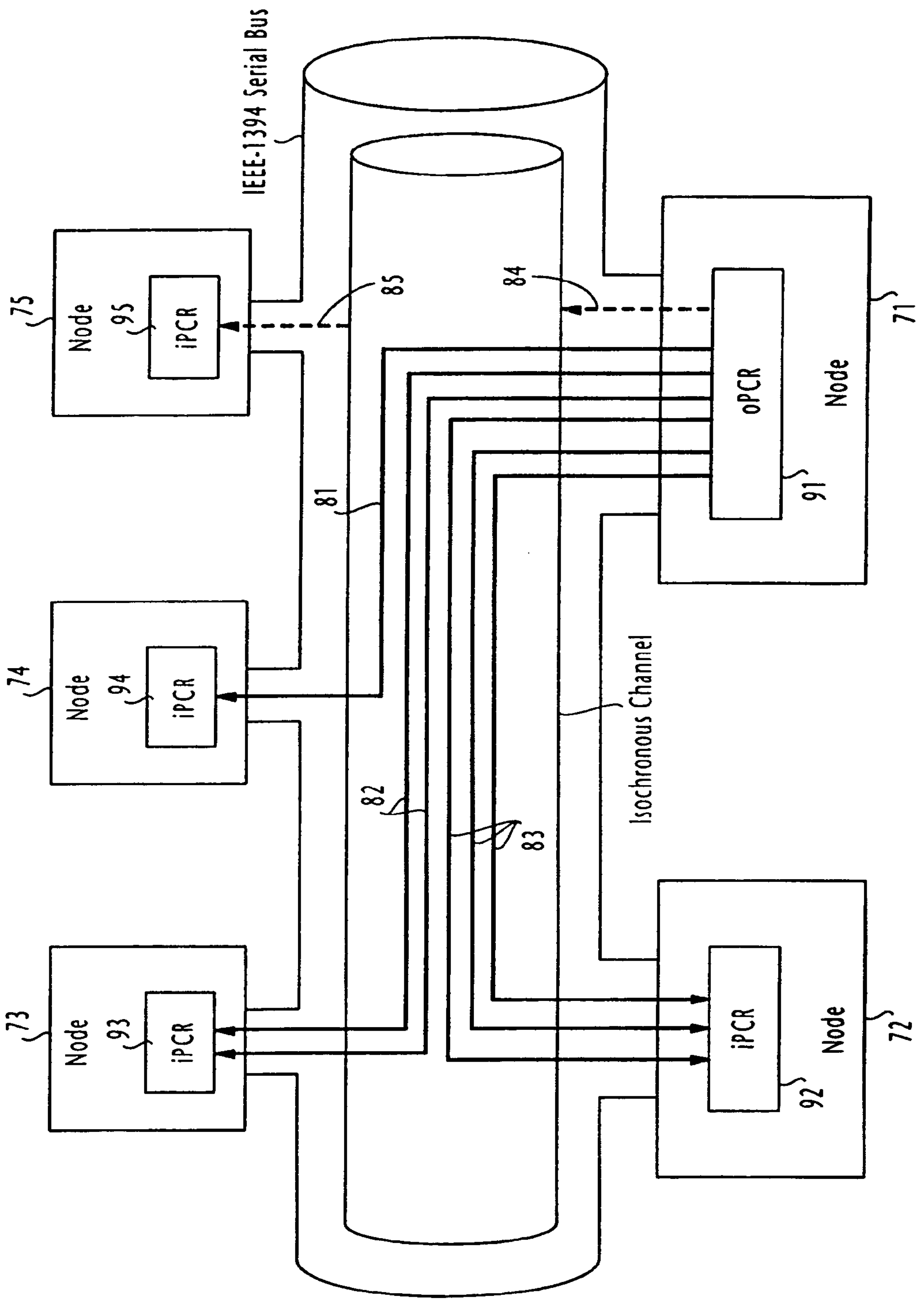
OUTPUT PLUG CONTROL REGISTER

Broadcast connection counter	Point-to-point connection counter	Channel number	Band-width	Overhead ID	Payload

INPUT PLUG CONTROL REGISTER

Broadcast connection counter	Point-to-point connection counter	Channel number	Reserved

FIG. 8


IEEE-1394 Serial Bus
Node
75
95
iPCR
85
84
71
Node
oPCR
91
81
Node
74
94
iPCR
Isochronous Channel
82
83
Node
73
93
iPCR
iPCR
Node
72
92

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 99 25 0019

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
D,Y	IEEE STANDARDS BOARD: "IEEE Standard for a High Performance Serial Bus" 1996 , IEEE COMPUTER SOCIETY , PISCATAWAY, US XP002102520 * page 199 * * page 203, paragraph 8.3.1.3 * * page 204, paragraph 8.3.1.5 - paragraph 8.3.1.6 * * page 221, paragraph 8.3.2.5 - page 223 * * page 236, paragraph 8.4.6 - paragraph 8.4.6.1 * * page 239, paragraph 8.5.2; figure 8.27 * ---	1-6	H04L12/24
Y	US 5 504 757 A (COOK ET AL) 2 April 1996 * column 5, line 35 - column 6, line 16 * * column 9, line 1 - column 10, line 13 * ---	1-6	
A	US 5 317 693 A (CUENOD ET AL) 31 May 1994 * column 9, line 20 - column 10, line 18; figures 1,2 * -----	1-6	
			TECHNICAL FIELDS SEARCHED (Int.Cl.6)
			H04L G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	11 May 1999	Gill, S

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 99 25 0019

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

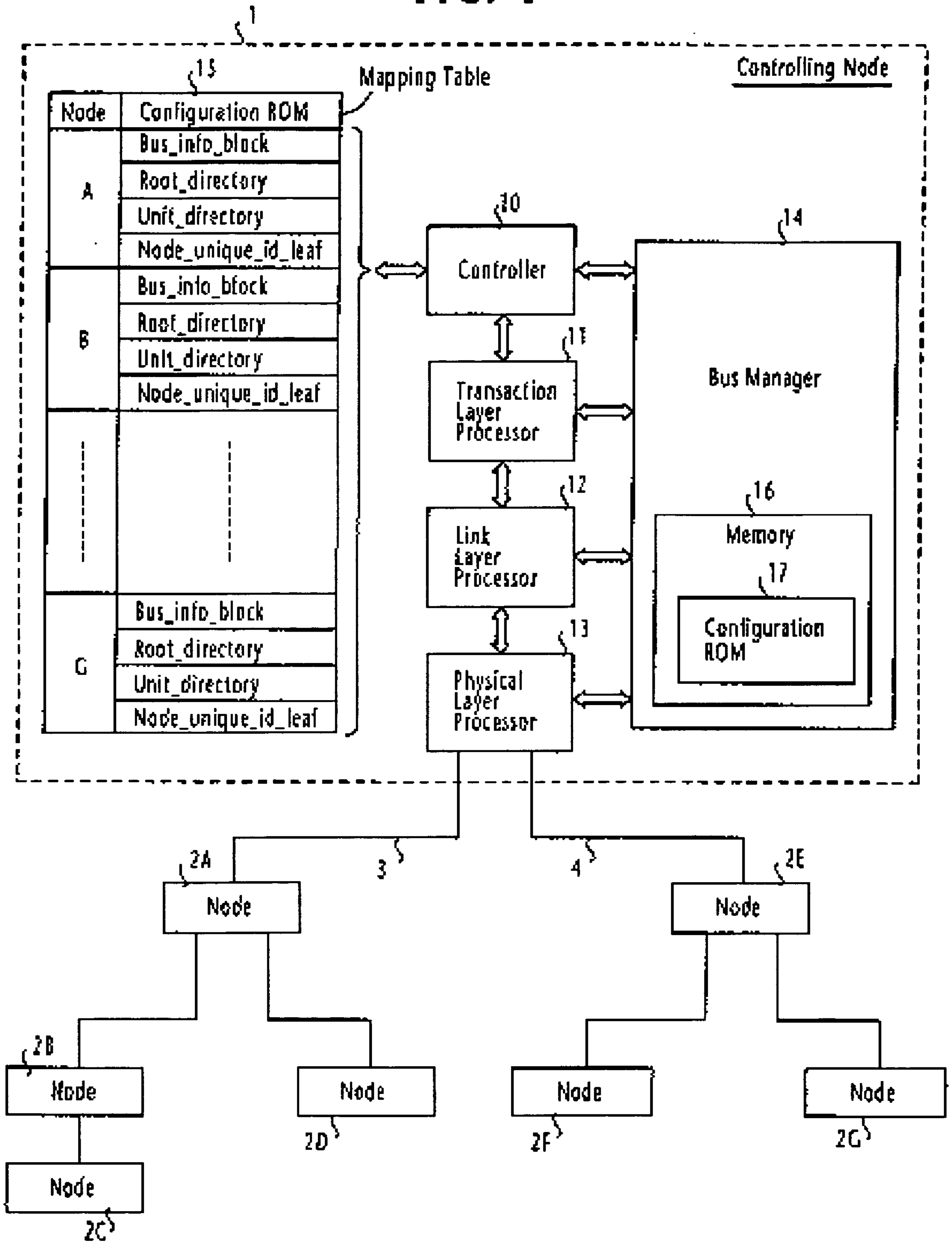
11-05-1999

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5504757 A	02-04-1996	NONE	
US 5317693 A	31-05-1994	NONE	

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

FIG. 1



1
Controlling Node
15
Mapping Table
Node
Configuration ROM
A
Bus_info_block
Root_directory
Unit_directory
Node_unique_id_leaf
B
Bus_info_block
Root_directory
Unit_directory
Node_unique_id_leaf
G
Bus_info_block
Root_directory
Unit_directory
Node_unique_id_leaf
10
Controller
11
Transaction Layer Processor
12
Link Layer Processor
13
Physical Layer Processor
14
Bus Manager
16
Memory
17
Configuration ROM
3
4
2A
Node
2E
Node
2B
Node
Node
2D
Node
2F
Node
2G
Node
2C

FIG. 2

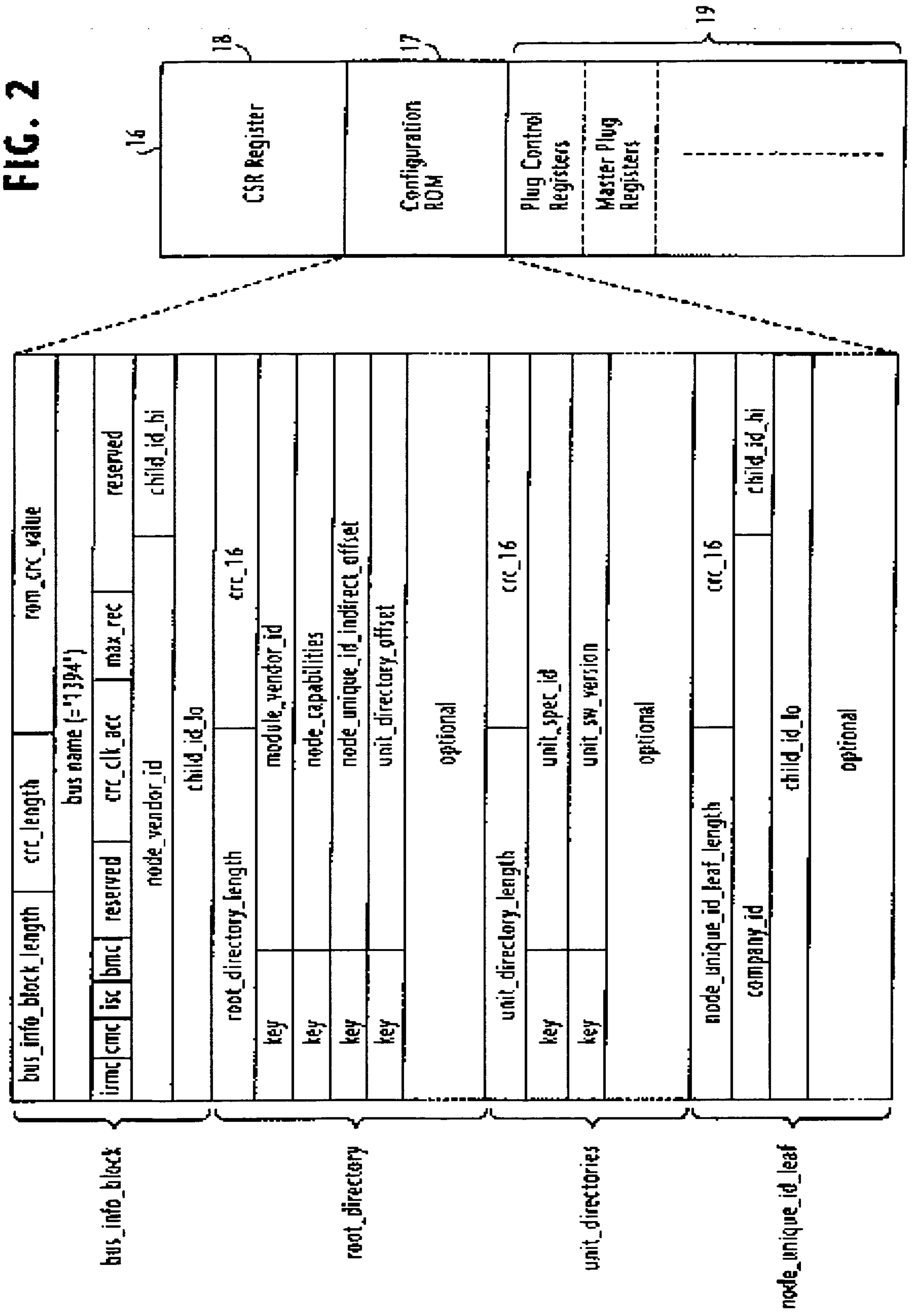

16
18
17
19
CSR Register
Configuration ROM
Plug Control Registers
Master Plug Registers
bus_info_block_length
crc_length
rom_crc_value
bus name (="1394")
irmc
cmc
isc
bmc
reserved
crc_clk_acc
max_rec
reserved
node_vendor_id
child_id_hi
child_id_lo
root_directory_length
crc_16
key
module_vendor_id
key
node_capabilities
key
node_unique_id_indirect_offset
key
unit_directory_offset
optional
unit_directory_length
crc_16
key
unit_spec_id
key
unit_sw_version
optional
node_unique_id_leaf_length
crc_16
company_id
child_id_hi
child_id_lo
optional
bus_info_block
root_directory
unit_directories
node_unique_id_leaf

FIG. 3

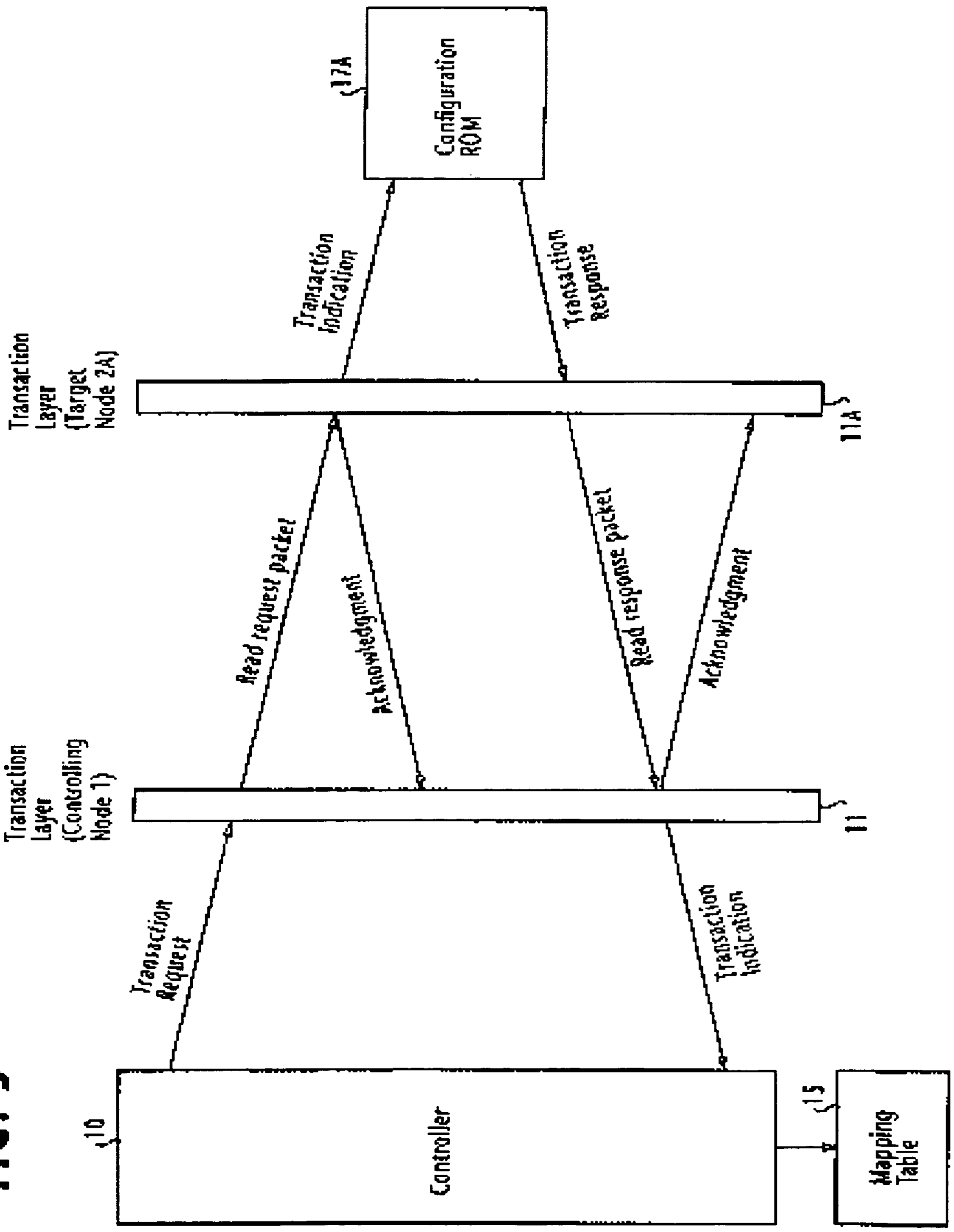

Transaction Layer (Controlling Node 1)
11
Transaction Layer (Target Node 2A)
11A
17A
Configuration ROM
Transaction Indication
Transaction Response
Read request packet
Acknowledgment
Read response packet
Acknowledgment
Transaction Request
Transaction Indication
10
Controller
15
Mapping Table

FIG. 4A

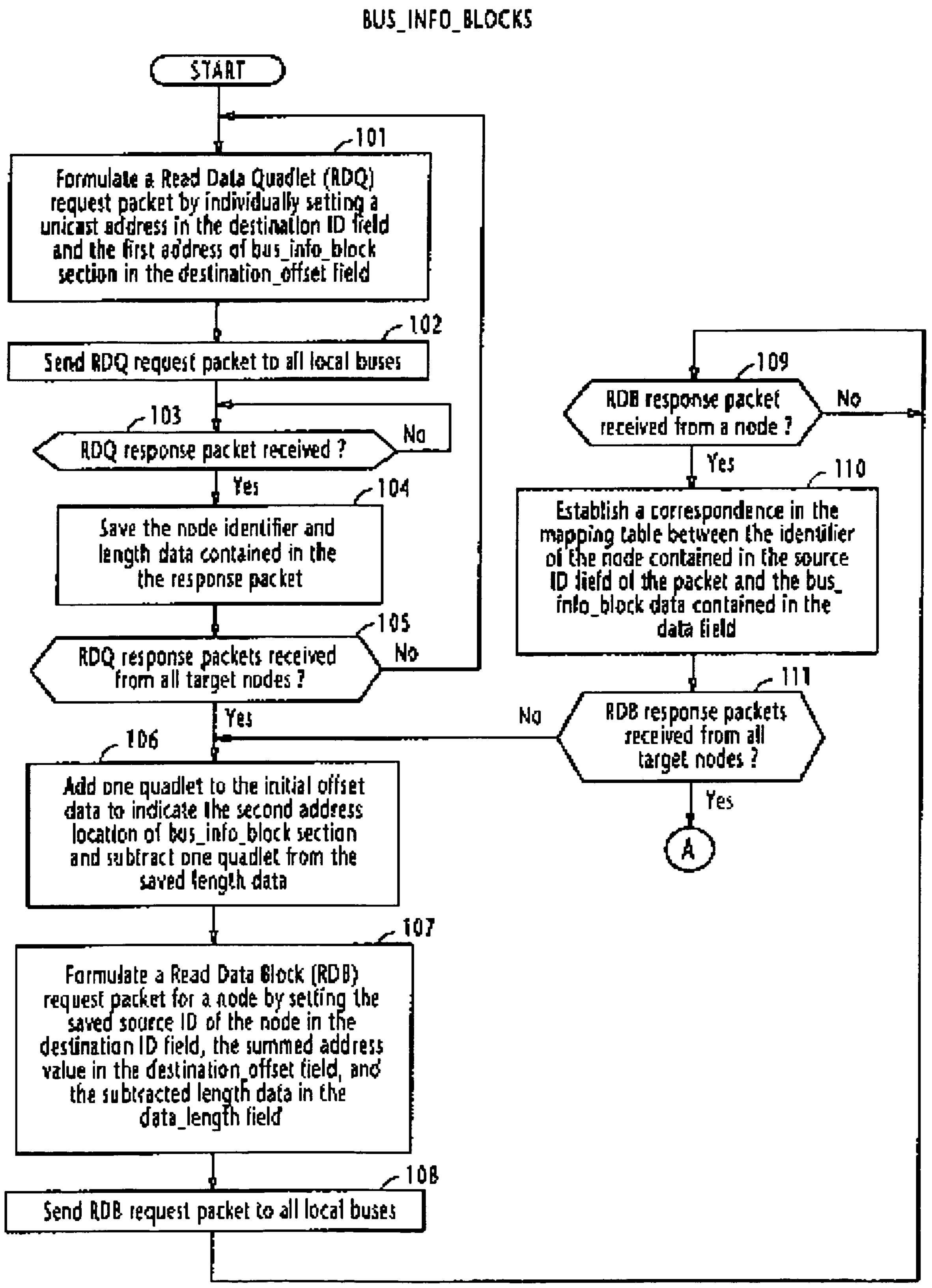

BUS_INFO_BLOCKS
START
101
Formulate a Read Data Quadlet (RDQ) request packet by individually setting a unicast address in the destination ID field and the first address of bus_info_block section in the destination_offset field
102
Send RDQ request packet to all local buses
103
RDQ response packet received ?
No
Yes
104
Save the node identifier and length data contained in the the response packet
105
RDQ response packets received from all target nodes ?
No
Yes
106
Add one quadlet to the initial offset data to indicate the second address location of bus_info_block section and subtract one quadlet from the saved length data
107
Formulate a Read Data Block (RDB) request packet for a node by setting the saved source ID of the node in the destination ID field, the summed address value in the destination_offset field, and the subtracted length data in the data_length field
108
Send RDB request packet to all local buses
109
RDB response packet received from a node ?
No
Yes
110
Establish a correspondence in the mapping table between the identifier of the node contained in the source ID field of the packet and the bus_info_block data contained in the data field
111
RDB response packets received from all target nodes ?
No
Yes
A

FIG. 4B

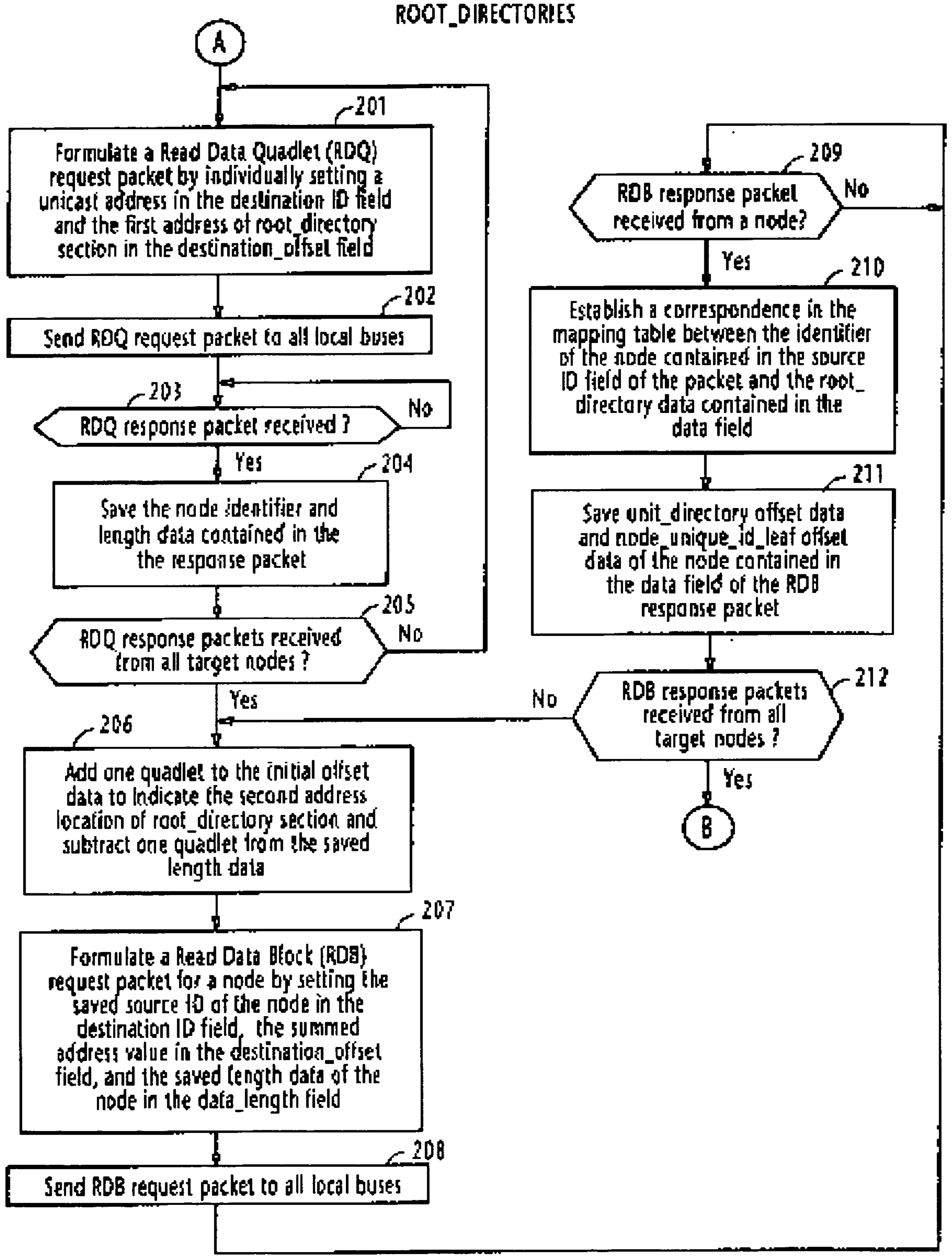

ROOT_DIRECTORIES
A
201
Formulate a Read Data Quadlet (RDQ) request packet by individually setting a unicast address in the destination ID field and the first address of root_directory section in the destination_offset field
202
Send RDQ request packet to all local buses
203
RDQ response packet received ?
No
Yes
204
Save the node identifier and length data contained in the the response packet
205
RDQ response packets received from all target nodes ?
No
Yes
206
Add one quadlet to the initial offset data to indicate the second address location of root_directory section and subtract one quadlet from the saved length data
207
Formulate a Read Data Block (RDB) request packet for a node by setting the saved source ID of the node in the destination ID field, the summed address value in the destination_offset field, and the saved length data of the node in the data_length field
208
Send RDB request packet to all local buses
209
RDB response packet received from a node?
No
Yes
210
Establish a correspondence in the mapping table between the identifier of the node contained in the source ID field of the packet and the root_directory data contained in the data field
211
Save unit_directory offset data and node_unique_id_leaf offset data of the node contained in the data field of the RDB response packet
212
RDB response packets received from all target nodes ?
No
Yes
B

FIG. 4C

UNIT_DIRECTORIES

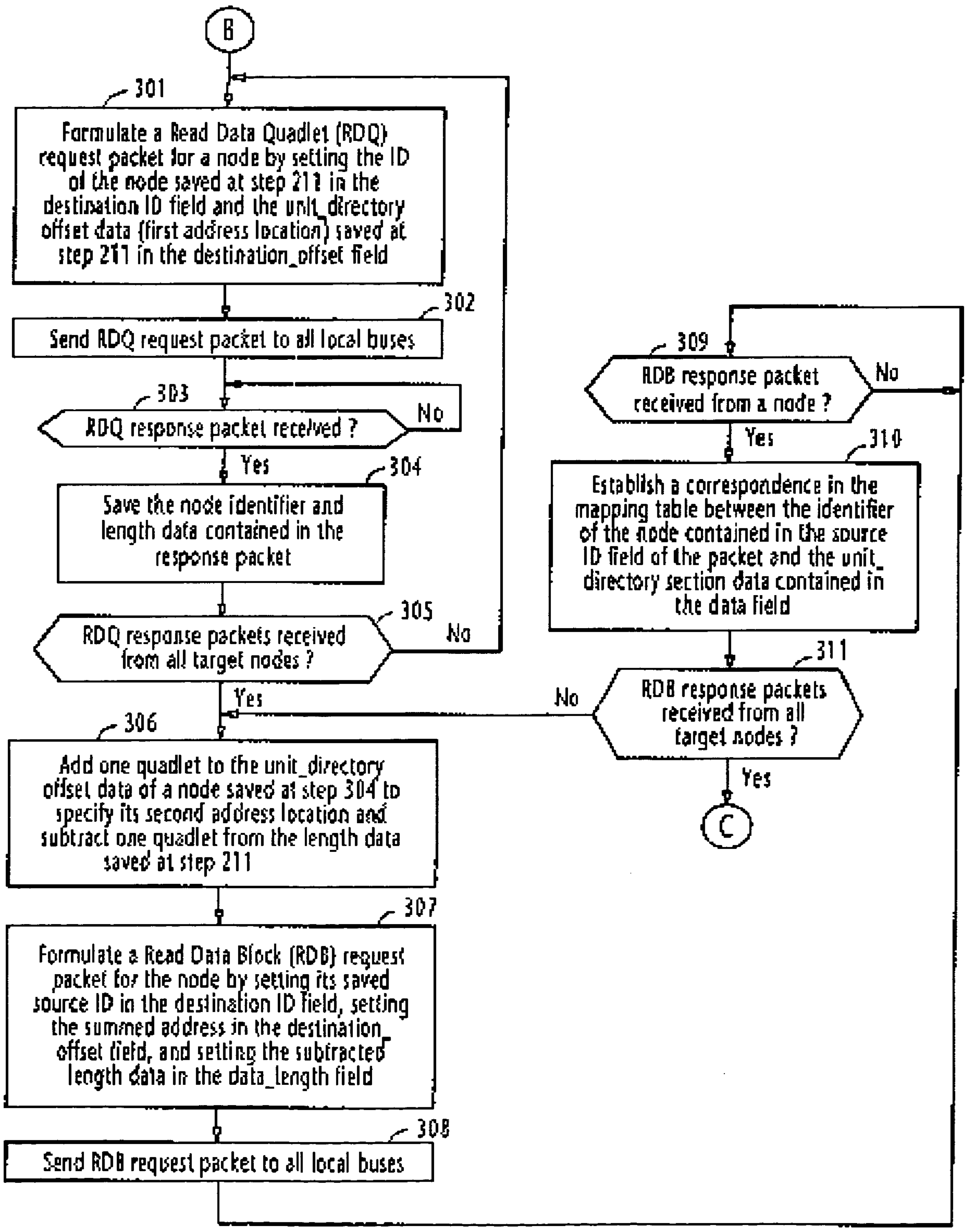

B
301
Formulate a Read Data Quadlet (RDQ) request packet for a node by setting the ID of the node saved at step 211 in the destination ID field and the unit_directory offset data (first address location) saved at step 211 in the destination_offset field
302
Send RDQ request packet to all local buses
303
RDQ response packet received ?
No
Yes
304
Save the node identifier and length data contained in the response packet
305
RDQ response packets received from all target nodes ?
No
Yes
306
Add one quadlet to the unit_directory offset data of a node saved at step 304 to specify its second address location and subtract one quadlet from the length data saved at step 211
307
Formulate a Read Data Block (RDB) request packet for the node by setting its saved source ID in the destination ID field, setting the summed address in the destination_offset field, and setting the subtracted length data in the data_length field
308
Send RDB request packet to all local buses
309
RDB response packet received from a node ?
No
Yes
310
Establish a correspondence in the mapping table between the identifier of the node contained in the source ID field of the packet and the unit_directory section data contained in the data field
311
RDB response packets received from all target nodes ?
No
Yes
C

FIG. 4D

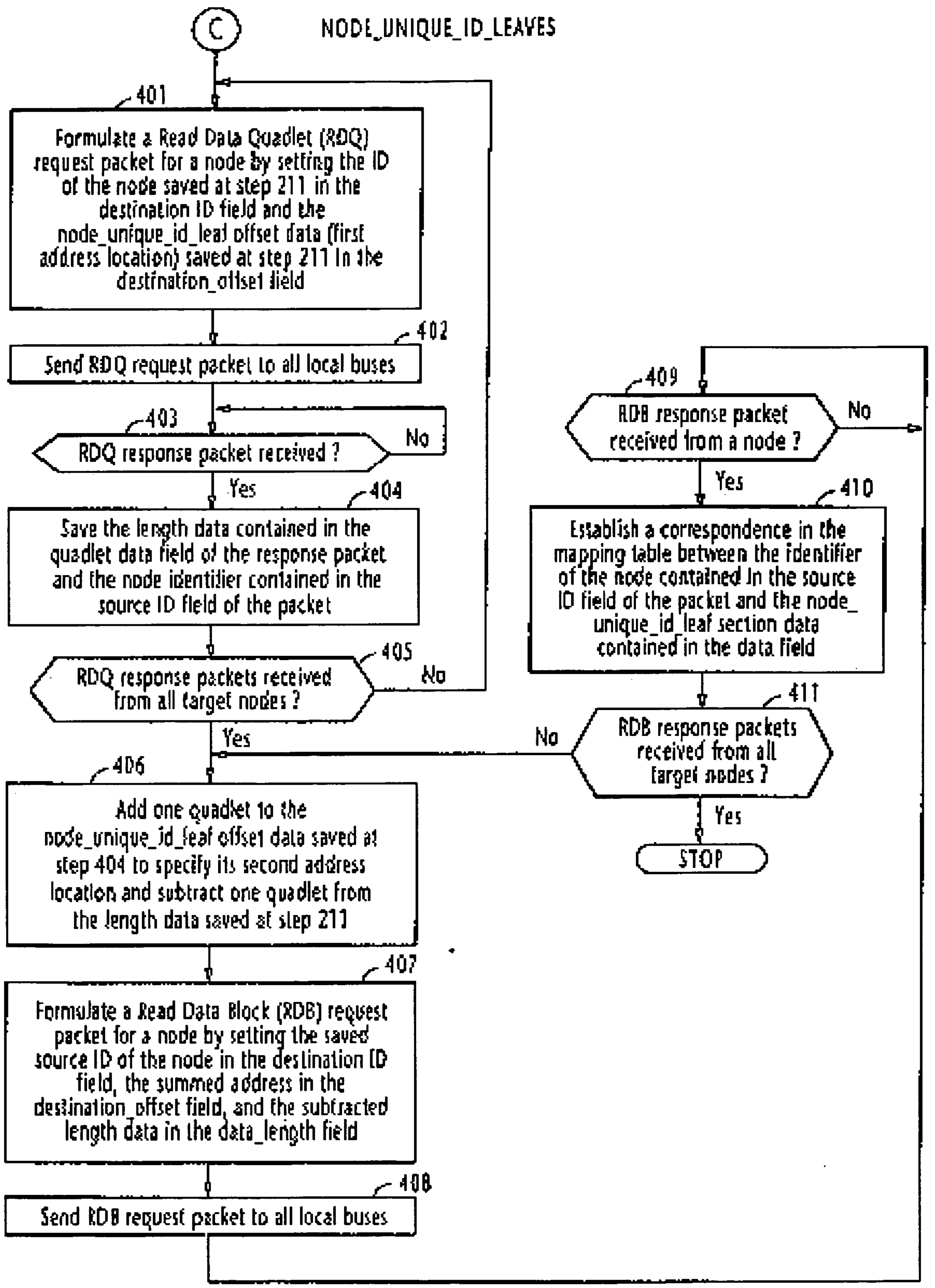

NODE_UNIQUE_ID_LEAVES
C
401
Formulate a Read Data Quadlet (RDQ) request packet for a node by setting the ID of the node saved at step 211 in the destination ID field and the node_unique_id_leaf offset data (first address location) saved at step 211 in the destination_offset field
402
Send RDQ request packet to all local buses
403
RDQ response packet received ?
No
Yes
404
Save the length data contained in the quadlet data field of the response packet and the node identifier contained in the source ID field of the packet
405
RDQ response packets received from all target nodes ?
No
Yes
406
Add one quadlet to the node_unique_id_leaf offset data saved at step 404 to specify its second address location and subtract one quadlet from the length data saved at step 211
407
Formulate a Read Data Block (RDB) request packet for a node by setting the saved source ID of the node in the destination ID field, the summed address in the destination_offset field, and the subtracted length data in the data_length field
408
Send RDB request packet to all local buses
409
RDB response packet received from a node ?
No
Yes
410
Establish a correspondence in the mapping table between the identifier of the node contained in the source ID field of the packet and the node_unique_id_leaf section data contained in the data field
411
RDB response packets received from all target nodes ?
No
Yes
STOP

FIG. 5A

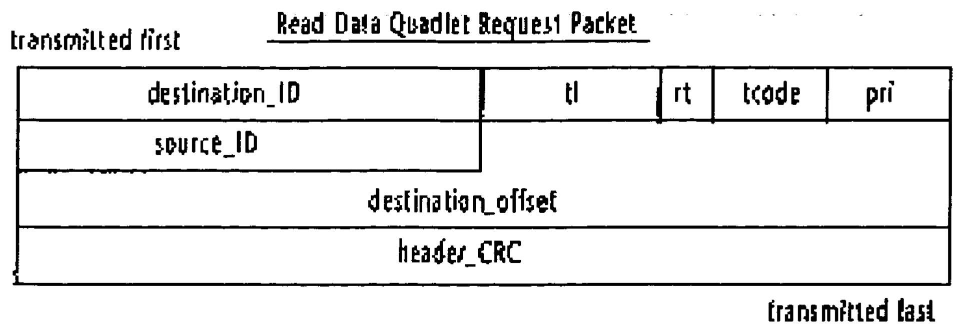


FIG. 5B

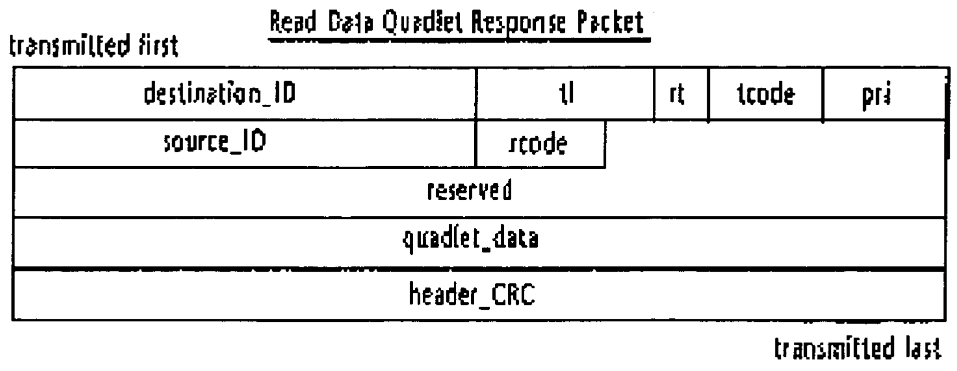


FIG. 5C

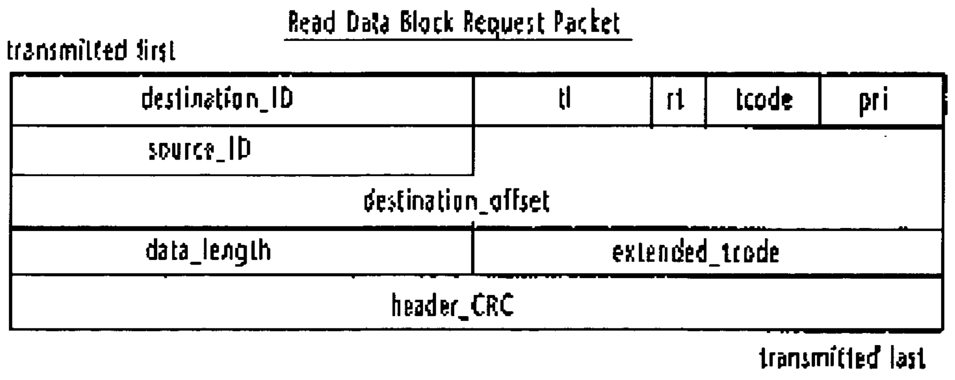


FIG. 5D

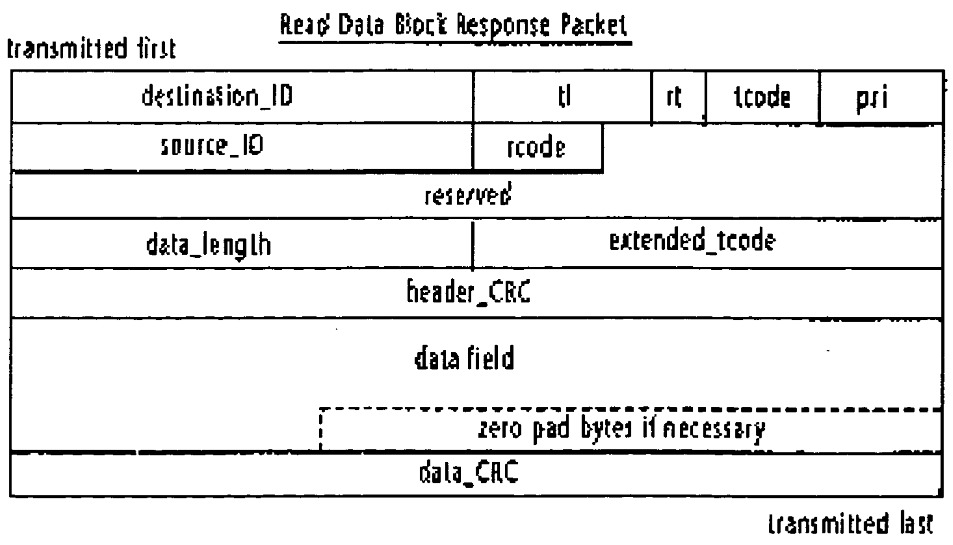


FIG. 6

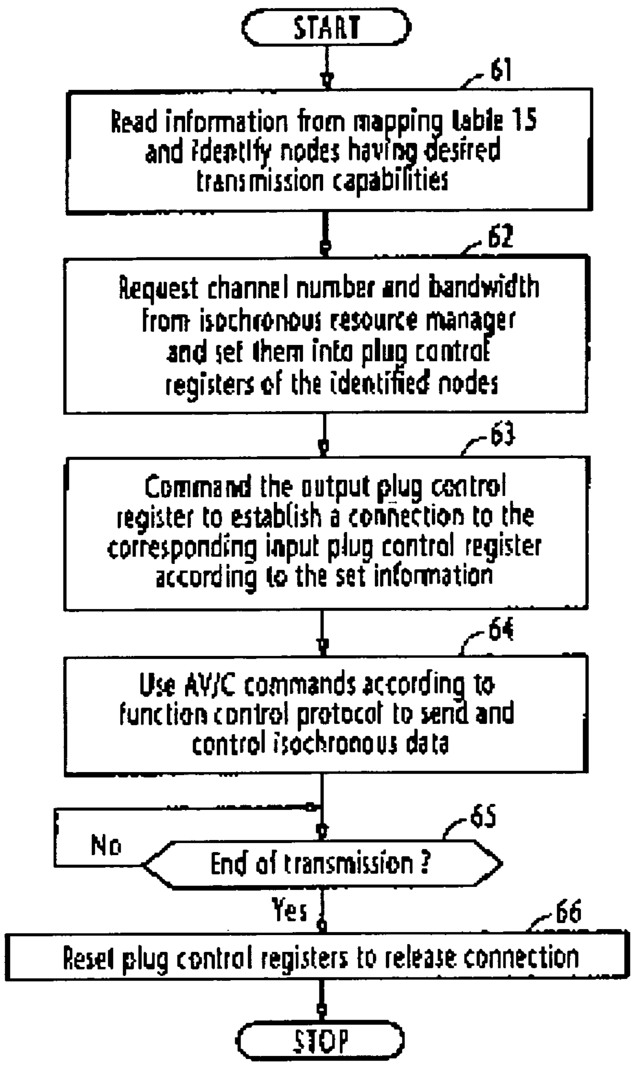

START
61
Read information from mapping table 15 and identify nodes having desired transmission capabilities
62
Request channel number and bandwidth from isochronous resource manager and set them into plug control registers of the identified nodes
63
Command the output plug control register to establish a connection to the corresponding input plug control register according to the set information
64
Use AV/C commands according to function control protocol to send and control isochronous data
65
End of transmission ?
No
Yes
66
Reset plug control registers to release connection
STOP

FIG. 7

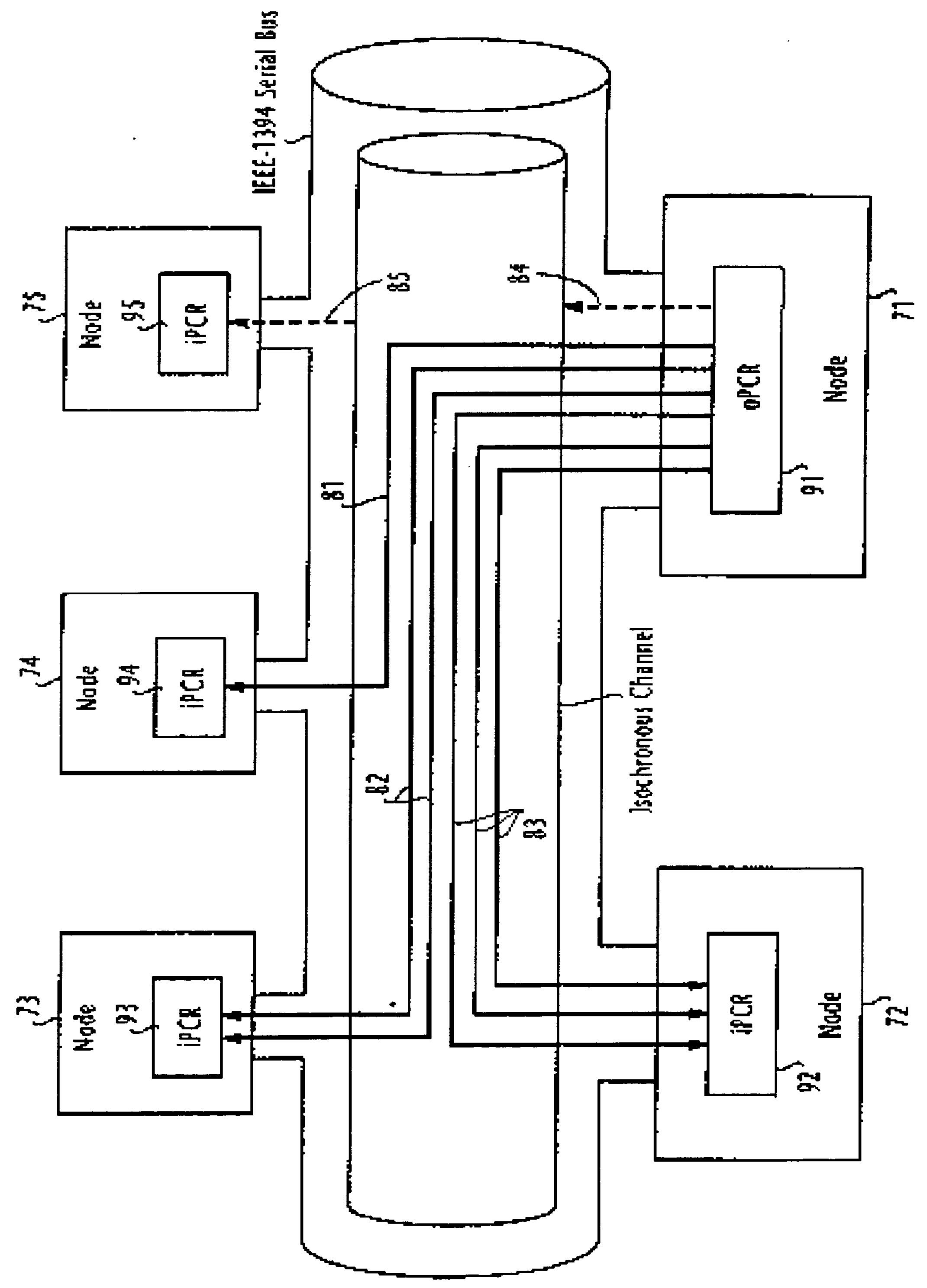
OUTPUT PLUG CONTROL REGISTER

Broadcast connection counter	Point-to-point connection counter	Channel number	Band-width	Overhead ID	Payload

INPUT PLUG CONTROL REGISTER

Broadcast connection counter	Point-to-point connection counter	Channel number	Reserved

FIG. 8


IEEE-1394 Serial Bus
Node
75
95
iPCR
85
84
Node
74
94
iPCR
81
82
83
Isochronous Channel
Node
73
93
iPCR
oPCR
Node
71
91
iPCR
Node
72
92